Numerical simulations of ion thruster-induced plasma dynamics

Von der Gemeinsamen Naturwissenschaftlichen Fakultät
der Technischen Universität Carolo-Wilhelmina
zu Braunschweig
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)
genehmigte
D i s s e r t a t i o n

von
Carsten Othmer

aus
Braunschweig

1. Referent:	Prof. Dr. K.-H. Glaßmeier
2. Referent:	Prof. Dr. U. Motschmann
eingereicht am:	24. September 2001
mündliche Prüfung (Disputation) am:	28. November 2001

2001

Vorveröffentlichungen der Dissertation

Teilergebnisse aus dieser Arbeit wurden mit Genehmigung der Gemeinsamen Naturwissenschaftlichen Fakultät, vertreten durch den Mentor der Arbeit, in folgenden Beiträgen vorab veröffentlicht:

Publikationen

Othmer, C., K. H. Glassmeier, U. Motschmann, J. Schüle, and Ch. Frick, Numerical simulations on shock-like ion thruster neutralization, *AIAA-Paper 2001-3784*, 2001.

Othmer, C., K. H. Glassmeier, U. Motschmann, and I. Richter, Numerical parameter studies of ion thruster beam neutralization, eingereicht bei *J. Prop. Power*, 2001.

Othmer, C., K. H. Glassmeier, U. Motschmann, and J. Schüle, Parallel PIC-code simulations of ion thruster-induced plasma dynamics, *Proc. ISSS-6*, 2001.

Othmer, C., and J. Schüle, Dynamic load-balancing of plasma particle-in-cell simulations: The taskfarm alternative, eingereicht bei *Comput. Phys. Comm.*, 2001.

Othmer, C., K. H. Glassmeier, U. Motschmann, J. Schüle, and Ch. Frick, Numerical simulation of ion thruster-induced plasma dynamics, eingereicht bei *Adv. Space Res.*, 2001.

Othmer, C., K. H. Glassmeier, U. Motschmann, J. Schüle, and Ch. Frick, Three-dimensional numerical simulations of ion thruster beam neutralization, *Phys. Plas.*, 7 (12), 5242, 2000.

Othmer, C., U. Motschmann, and K. H. Glassmeier, Creation of spatial charge separation in plasmas with rigorously charge-conserving local electromagnetic field solvers, eingereicht bei *J. Comput. Phys.*, 2000.

Tagungsbeiträge

Othmer, C., K. H. Glassmeier, U. Motschmann, and J. Schüle, Parallel PIC-code simulations of ion thruster-induced plasma dynamics, *6th International Symposium for Space Plasma Simulations*, Garching, 2001.

Othmer, C., K. H. Glassmeier, U. Motschmann, J. Schüle, and Ch. Frick, Numerical simulations of ion thruster beam neutralization, *37th Joint Propulsion Conference*, Salt Lake City, USA, 2001.

Othmer, C., and G. J. Pringle, Particle-in-cell simulations of ion thrusters, *TAM User Group Meeting*, Edinburgh, Scotland, 2001.

Othmer, C., J. Schüle, Dynamic load-balancing of plasma particle-in-cell simulations: The taskfarm alternative, *Conference on Computational Physics*, Aachen, 2001.

Vorveröffentlichungen der Dissertation (Forts.)

Othmer, C., K. H. Glassmeier, U. Motschmann, J. Schüle, and Ch. Frick, Simulations of the plasma dynamics in the surroundings of an ion thruster, *33rd COSPAR Scientific Assembly*, Warschau, Polen, 2000.

Othmer, C., K. H. Glaßmeier, U. Motschmann, J. Schüle, and Ch. Frick, Numerische Simulationen zur Plasmadynamik in der Umgebung eines Ionentriebwerks, *Frühjahrstagung der DPG*, Bremen, 2000.

Contents

Chapter 1

Ion thrusters and their plasma environment

Electric propulsion devices are regarded as the next-generation type of space propulsion for interplanetary missions. Recently, interest in such devices has grown considerably: NASA's Deep Space 1 (DS1) spacecraft, which is currently on its way to encounter comet Borelly, is the first interplanetary mission that is equipped with an ion thruster, and the European Space Agency is planning to employ an electric propulsion engine on its SMART-1 mission to the Moon.

Fig. 1.1 shows a photograph of the ion thruster that operates on Deep Space 1. It has a diameter of about 40 cm and, as every ion engine, it derives its thrust from the electrostatic acceleration of heavy ions. The noble gas xenon, which serves as the propellant aboard DS1, flows into a kind of ionization chamber (Fig. 1.2),

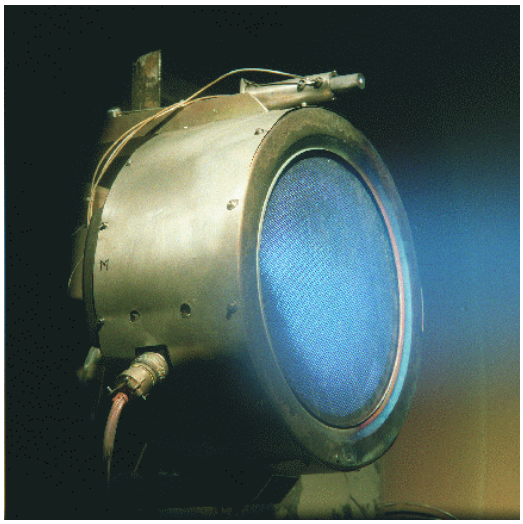

Figure 1.1: Photograph of the DS1 ion thruster. Note the hollow-cathode neutralizer attached at the top. The thruster diameter is approximately 40 cm.

where it is ionized via electron impact. The ions are extracted from the chamber by a grid system with an applied voltage in the order of 1 kV. Outside the chamber, they form a dense beam travelling at 35 km/s. This ejection velocity is about one order of magnitude higher than those of conventional chemical propulsion systems.

The maximum thrust thereby achieved amounts to 90 mN in the case of DS1. A total of 80 kg of xenon is ejected during the 6000 hours of operation, which results in an overall velocity change Δv of 4.2 km/s for the 500 kg spacecraft. About 10 times more fuel would be required to obtain the same velocity gain with a conventional chemical propulsion system. It is this mass efficiency that makes electric propulsion a very attractive option for long-term missions.

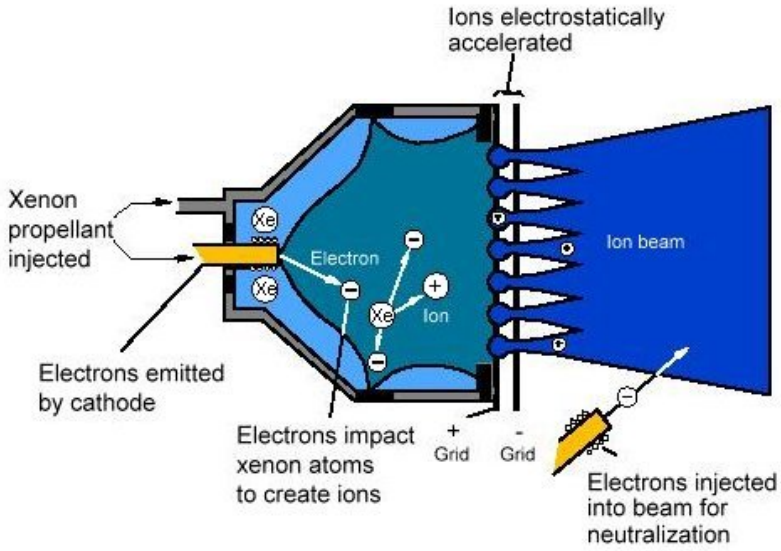

Figure 1.2: A schematic cut through the DS1 ion engine showing the ionization chamber, the acceleration grid and the hollow-cathode neutralizer outside the chamber.

In order for the ion beam to propagate, it needs to be electrically neutralized. For this purpose, a hollow-cathode is mounted on the thruster periphery that injects electrons into the beam. The electrostatic forces between electrons and ions are thought to induce a mixing between these two species and to generate a quasi-neutral plasma beam. Recombination is negligible in the neutralization process due to its small effective cross section. Apart from that, some unionized neutrals escape from the thruster and form a neutral cloud directly behind the thruster exit. Through collisions with energetic beam ions, these neutrals generate a low-energy charge-exchange plasma plume around the thruster (CEX).

Hence, an ion thruster produces a very dynamic plasma environment. Three different plasma regimes can be identified: (1) the CEX plasma plume around the spacecraft, (2) the neutralization, i. e. the mixing between the beam ions and the

electrons ejected by the hollow-cathode, and (3) the interaction of the neutralized thruster beam with the ambient solar wind.

Apart from being a very interesting active plasma experiment in itself, the ion thruster-induced plasma environment has raised various concerns in terms of its potential impact on the spacecraft and on scientific instruments aboard. Of special relevance in this respect are the CEX ions: They flow back to the spacecraft and cause significant erosion of the thruster grid system, which is actually the main lifetime-limiting process for ion thrusters [Tartz et al., 1999]. Moreover, plasma waves, which are excited in the neutralization region or via interactions between the thruster beam and the solar wind, may induce electromagnetic interference.

While the dynamics of CEX ions have been the subject of extensive theoretical studies [Wang and Brophy, 1995; Samanta Roy et al., 1996a, 1996b; Wang et al., 1996] and some experimental verifications [de Boer, 1997; Tartz et al., 1999; Wang et al., 2000], the neutralization process and the solar wind-thruster beam interaction have received only minor attention. The most thorough treatments of ion thruster beam neutralization date back to as early as the 1960s. Buneman and Kooyers [1963], Dunn and Ho [1963], and Wadhwa et al. [1965] carried out one- and two-dimensional computer simulations of the mixing between cold streaming ions and cold or hot electrons. These authors found that self-excited fluctuating space-charge fields at the electron plasma frequency provide the mixing of electrons and ions, and lead to the creation of a stably streaming plasma.

Given the very limited computational possibilities of that time, these investigations were primarily aimed at a “proof of principle” of ion beam neutralization via electron injection, rather than dedicated to an in-depth study of the plasma physical processes that accompany the neutralization. The main questions that are still to be tackled concerning ion thruster beam neutralization are:

- Does the neutralization lead to a plasma in thermal equilibrium?
- If yes, which is the mechanism that provides thermalization?
- What does the thermalized state of the plasma look like in terms of e. g. temperature and degree of neutralization?
- Are there any plasma instabilities occurring upon neutralization?
- How does the neutralization process depend on basic parameters as beam width, injection velocities, ambient magnetic field?

The question of interaction between the quasi-neutral beam plasma and the ambient solar wind has not been considered at all. It differs considerably from other beam-plasma scenarios in that the high-density beam has a lateral scale of just several decimeters, which is much less than typical scale lengths of the solar

wind. While the thruster beam will pass rather unaffected through the tenuous solar wind due to the huge density contrast between the beam and the solar wind (roughly $4 \cdot 10^9$), it might, however, drive instabilities in the solar wind plasma.

To develop a numerical simulation that allows to investigate both the ion thruster neutralization regime and the possible interaction of the neutralized ion beam with the ambient solar wind is one of the two aims of this work, the other being the application of this simulation for an in-depth-study of ion thruster beam neutralization. The distinct characters of both scenarios had to be taken into account when designing the numerical model. How this was accomplished by setting up the three-dimensional electromagnetic particle-in-cell simulation "ISOLDE" is shown in Ch. 2, which describes the development of the simulation code.

As our simulation has high requirements towards computational performance, the code had to be implemented on a parallel computer (Ch. 3). In a straightforward manner, we first develop a parallelized version of the code that runs fairly efficiently on up to 8 processors. In a second step, we apply a more sophisticated parallelization strategy in order to run the code with high efficiency on a massively parallel computer.

The remaining chapter is devoted to the application of the simulation code to the problem of ion thruster beam neutralization. We start our investigations of the neutralization scenario with a quasi-one-dimensional geometry, where electrons and ions are injected into the simulation volume through the same orifice. After addressing the influence of an axial magnetic field on the neutralization process, we apply our simulation to a fully three-dimensional configuration with spatially separated electron and ion sources. Against the background of our findings, we finally analyse some actual measurements of the DS1 ion thruster environment.

Our simulation results will not only provide substantial insight into the plasma physical processes of ion thruster beam neutralization, but also allow us to give concrete recommendations for an optimized thruster design. We will present the criteria to be met in order for future ion thrusters to benefit from a more efficient neutralization process with less erosion through CEX ions and less electromagnetic interference.

Chapter 2

The numerical model

Computer simulations have become a standard tool in space plasma and laboratory plasma research. Setting up a numerical plasma simulation means finding adequate numerical representations of (a) the involved electric and magnetic fields and of (b) the plasma components. For the fields, one generally has the choice between an electrostatic and an electromagnetic simulation, while the plasma itself can be treated as a fluid or as an ensemble of many particles.

The neutralization, i. e. the mixing between initially spatially separated electrons and ions, involves a highly non-neutral plasma and is obviously dominated by electrostatic forces. Once the beam is neutralized, however, the electrostatic forces become less important, and electromagnetic phenomena such as the self-consistently generated magnetic field of the beam particles might play a role, e. g. for the possible interaction of the beam with the solar wind.

An electrostatic simulation produces per definition a curl-free electric field. Since the temporal change of the magnetic field is determined by the curl of $\mathbf{E}$ via

$$\frac{\partial \mathbf{B}}{\partial t} = \frac{\partial (\mathbf{B}_0 + \mathbf{b})}{\partial t} = \frac{\partial \mathbf{b}}{\partial t} = -\nabla \times \mathbf{E}\,, \tag{2.1}$$

where $\mathbf{B}_0$ is some constant background field, electrostatic models cannot account for a self-consistently generated magnetic field $\mathbf{b}$. This field influences the plasma dynamics e. g. via the force $\mathbf{F}$ that it exerts on the particles:

$$\mathbf{F} = q\left[\mathbf{E} + \mathbf{v} \times (\mathbf{B}_0 + \mathbf{b})\right]. \tag{2.2}$$

Neglecting $\mathbf{b}$ is therefore a justified simplification, as long as it is much smaller than the background magnetic field $\mathbf{B}_0$. This is, however, not necessarily the case for a neutralized ion thruster beam. The self-consistently generated magnetic field of the thruster beam can readily be calculated via Biot-Savart's law from Table 1, which summarizes some of the DS1 ion engine parameters. Even for a 99% neutralized ion beam, it amounts to about 25 nT at the beam surface, as compared to the 5 nT of the solar wind at 1 AU. Hence, a proper description of this plasma scenario requires a fully electromagnetic simulation model.

Ion bulk velocity	v_{i0}	$= 3.5 \cdot 10^4 \mathrm{m/s}$
Electron thermal velocity	v_{e0}^{th}	$= 4 - 9 \cdot 10^5 \mathrm{m/s}$
Injection velocity ratio	η	$= v_{e0}^{th}/v_{i0} = 12 - 25$
Ion beam density	n_{i0}	$= 4 \cdot 10^{15} \mathrm{m}^{-3}$
Electron plasma frequency	ω_{pe}^0	$= 3.7 \cdot 10^9 \mathrm{s}^{-1}$ (for $n_e = n_{i0}$)
Electron Debye length	λ_D	$= 2 \cdot 10^{-4} \mathrm{m}$
Electron inertia length	l_e	$= 8 \cdot 10^{-2} \mathrm{m}$
Ion beam diameter	b	$= 0.3 \mathrm{m} = 3.75 l_e = 1500 \lambda_D$

Table 1: Some basic DS1 parameters [after Wang et al., 2000].

For the numerical representation of the plasma components themselves there are two options: either a fluidal or a corpuscular description [e. g. Winske and Omidi, 1996]. Within a fluidal approach, each volume element of a plasma component is assigned a density and a *single* velocity. Such a model does not account for the thermal velocity *distribution* of the plasma particles. Therefore, it cannot describe kinetic effects, i. e. effects that have their origin in velocity space. It is a valid approximation as long as typical spatial and temporal scales of the simulation regime greatly exceed the single-particle scales such as inertia length or gyroperiod.

Within a corpuscular description, a plasma is modeled as hundreds of thousands of test particles. Such a model follows the evolution of the particle orbits in a self-consistently determined electromagnetic field. As it allows to correctly model the velocity distribution of the plasma particles, its great advantage as compared to a fluidal approach is the inclusion of kinetic effects. However, even with today's supercomputers, there is a great discrepancy between the number of particles that can be handled computationally ($10^7 - 10^9$) and the size of a realistic physical system (typically 10^{22} particles). The actual plasma particles have therefore to be grouped to "macro-particles" or "super-particles".

Due to the great difference in mass between electrons and ions, the spatial and temporal scales of these two plasma components are usually widely separated. If the plasma processes of interest have scales that are close to e. g. the ion scales, one might want to include the kinetic effects on the ion scale via a particle representation of the ions, while the details of the electron dynamics might not be of importance. In this case, a fluidal description of the electrons might suffice and would lead to a significant reduction of the numerical workload for the electrons. Such a mixture of fluidal and corpuscular description is realized in so-called hybrid simulations.

The basic questions of ion thruster beam neutralization that were stated in the introduction, address, among other things, the mechanism by virtue of which a thermalization of the plasma takes place and also the possibility of instabilities occurring upon neutralization. As these processes involve wave-particle interactions, probably on both the ion and the electron scale, the only way of implementing them into the simulation is via a particle simulation.

Hence, an adequate simulation of both the neutralization process and the interaction of the neutralized beam with the solar wind requires an electromagnetic particle model, which is, moreover, capable of dealing with highly non-neutral plasmas. In order to reproduce the actual geometric configuration of an ion thruster with spatially separated electron and ion sources, the code has to be three-dimensional. As in terms of computer performance, such a 3D particle code has very high requirements, it will have to be implemented on a parallel computer. This has to be kept in mind during the design of the code.

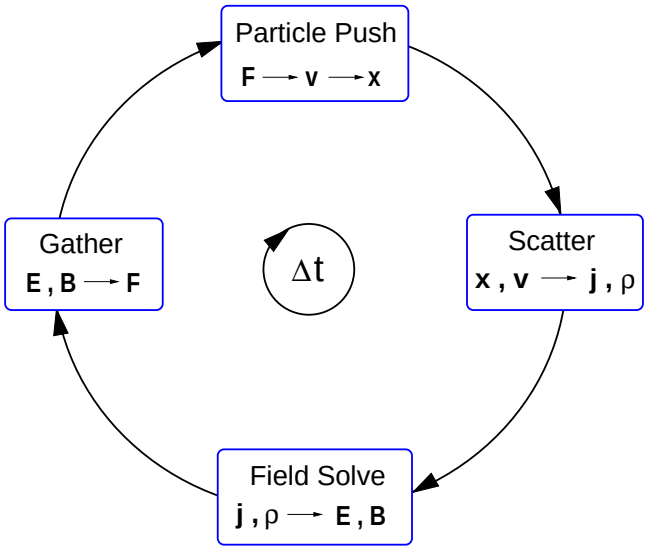

Figure 2.1: The computational four-step cycle of a particle simulation [modified after Birdsall and Langdon, 1985].

In essence, a particle simulation consists of a loop of four steps that is run through once per time step (Fig. 2.1):

1. Particle positions and velocities are updated in the **particle push** by integrating the equations of motion.
2. In the **scatter** step, the new charge and / or current density is calculated from the new particle positions and velocities. Since the particles can be located anywhere within the simulation domain, i. e. have continuous coordinates, but the macroscopic field quantities are defined only on discrete grid points, the charge and current carried by the particles have to be deposited (“scattered”) on the grid points.

3. The new charge and current density enter the **field solve** routine to update the electromagnetic fields.

4. The **gather** step calculates the force that is acting on each particle by interpolating ("gathering") the electromagnetic fields from the grid points to the particle positions.

In the following, we will describe the numerical realization of each of these steps and also of appropriate initial and boundary conditions for fields and particles.

2.1 Field solve

Electromagnetic simulation codes have to solve the full set of Maxwell's equations:

$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E} \tag{2.3}$$

$$\frac{1}{c^2}\frac{\partial \mathbf{E}}{\partial t} = \nabla \times \mathbf{B} - \mu_0 \mathbf{j} \tag{2.4}$$

$$\nabla \cdot \mathbf{E} = \rho/\varepsilon_0 \tag{2.5}$$

$$\nabla \cdot \mathbf{B} = 0\,. \tag{2.6}$$

Two of them, namely (2.3) and (2.4), have already a form that is ideally suited for numerical integration of the time development of $\mathbf{E}$ and $\mathbf{B}$: Provided that the electric field $\mathbf{E}(\mathbf{x})$ and the magnetic field $\mathbf{B}(\mathbf{x})$ are given for all $\mathbf{x}$ at a certain time, Eqn. (2.3) allows to compute the "new" magnetic field $\mathbf{B} + \Delta t \cdot \partial_t \mathbf{B}$ for all $\mathbf{x}$. In order to obtain the new electric field $\mathbf{E}$, the updated $\mathbf{B}$-field and the current density $\mathbf{j}$, which is provided by the scatter routine (Sec. 2.2), are substituted into Eqn. (2.4). The updated $\mathbf{E}$-field is then plugged into Eqn. (2.3) again, and so on.

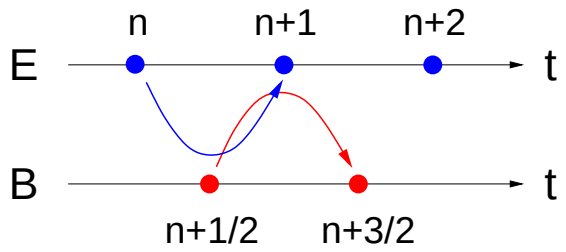

Figure 2.2: The leap-frog method: Electric and magnetic field are shifted half a time step apart.

In order to make this method second order accurate, central differences *in time* are used for the integration. This requires to shift $\mathbf{E}$ and $\mathbf{B}$ half a time step apart on the time axis (Fig. 2.2). Electric and magnetic field are defined on full integer

and half integer time steps, respectively. Hence, the numerical time integration of $\mathbf{E}$ and $\mathbf{B}$ reads as follows:

$$\mathbf{B}^{n+1/2} = \mathbf{B}^{n-1/2} - \Delta t \cdot \nabla \times \mathbf{E}^{n} \tag{2.7}$$

$$\mathbf{E}^{n+1} = \mathbf{E}^{n} + c^2 \Delta t \cdot \left(\nabla \times \mathbf{B}^{n+1/2} - \mu_0 \mathbf{j}^{n+1/2} \right) , \tag{2.8}$$

where the superscripts denote the time level. For obvious reasons, this explicit integration method, which is widely used in plasma simulations [e. g. Birdsall and Langdon, 1985; Birdsall, 1991; Buneman, 1993; Wang et al., 1995], is called "leap-frog".

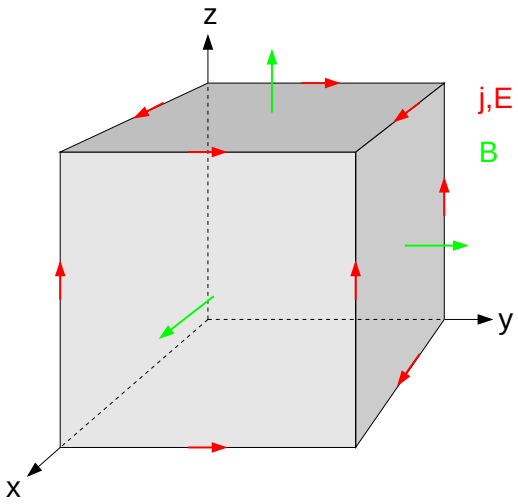

Figure 2.3: The Yee lattice.

For the computation of the curl of electric and magnetic field to enter Eqns. (2.7) and (2.8), $\mathbf{E}$ and $\mathbf{B}$ have to be defined on a spatial grid. The kind of grid we employ for our simulation is the so-called "Yee lattice", which was introduced by Yee [1966] precisely for the integration of Maxwell's equations. It consists of a 3D cubic grid with grid spacings $\Delta X = \Delta Y = \Delta Z$ in which $\mathbf{E}$ and $\mathbf{j}$ are defined at mid-points of cell-edges while the $\mathbf{B}$ components are defined at midpoints of cell-surfaces (Fig. 2.3). In the complementary grid, i. e. the mesh with cell corners and cell centres interchanged, the situation is reversed: $\mathbf{B}$ is located at mid-points of cell-edges while $\mathbf{E}$ and $\mathbf{j}$ are defined at midpoints of cell-surfaces. This ensures that the field component data are exactly available where needed for updating the fields and allows to use the most physical method of field-processing, namely updating the fluxes of $\mathbf{E}$ ($\mathbf{B}$) through any cell face from the circulation of $\mathbf{B}$ ($\mathbf{E}$) around those faces, with charge flux to be included in the case of $\mathbf{E}$.

As an example, we illustrate the update of B_z in Fig. 2.4. Considering the respective locations of $\mathbf{B}$ and $\mathbf{E}$ in the Yee grid, the z component of Eqn. (2.7) reads

$$B_z^{n+1/2}(i,j,k) = B_z^{n-1/2}(i,j,k) + \frac{\Delta t}{\Delta X} \cdot \Big[\left(E_y^n(i+1,j,k) - E_y^n(i,j,k) \right) - \left(E_x^n(i,j+1,k) - E_x^n(i,j,k) \right) \Big] . \tag{2.9}$$

The curl operator is realized via central differences. The integration *in space* is therefore second order accurate, much as is the leap-frog method for the time integration. Hence, leap-frogging $\mathbf{E}$ and $\mathbf{B}$ in the Yee lattice guarantees overall second order accuracy for the field update.

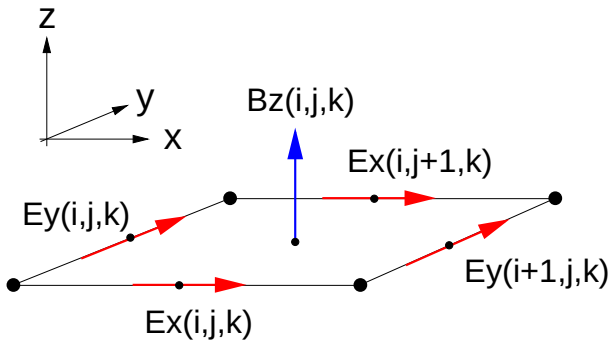

Figure 2.4: Field integration in the Yee lattice: The update of B_z via the curl of the electric field according to Eqn. (2.9).

From what we have shown so far, it seems that Maxwell's two curl equations already suffice to determine the temporal evolution of the electromagnetic field. However, to some extent, also the second pair of Maxwell's equations, (2.5) and (2.6), have to be taken care of. This will be demonstrated in the following.

We differentiate Eqn. (2.6) with respect to time, change the order of differentiation, substitute Eqn. (2.3), which is used to update $\mathbf{B}$ in the integration scheme described above, and end up with

$$\frac{\partial}{\partial t}(\nabla \cdot \mathbf{B}) = \nabla \cdot \frac{\partial \mathbf{B}}{\partial t} = -\nabla \cdot (\nabla \times \mathbf{E}). \tag{2.10}$$

Since for *differential* quotients

$$\nabla \cdot \nabla \times \ldots \equiv 0\,, \tag{2.11}$$

the right hand side of Eqn. (2.10) vanishes identically, meaning that $\nabla \cdot \mathbf{B}$ remains zero if it was so initially. If a divergence of a curl also vanishes for *finite* differences is not a trivial question, but depends on the discretized representation of the $\nabla\cdot$ and $\nabla\times$ operators. That this is the case for central differences in the Yee grid can, however, readily be checked. Hence, our integration scheme ensures that the $\mathbf{B}$ field remains divergenceless if it is so at $t = 0$, which can easily be accomplished by e. g. $\mathbf{B}|_{t=0}$ =const.

While the divergence equation for $\mathbf{B}$ (2.6) can be fulfilled in a straightforward manner by choosing appropriate initial magnetic field configurations, the treatment of Eqn. (2.5) for the divergence of $\mathbf{E}$ (Gauss' law) requires more caution: Differentiating it with respect to time, changing the order of differentiation and

employing Eqn. (2.4) to substitute $\partial_t \mathbf{E}$ leads to the charge continuity equation:

$$\frac{\partial}{\partial t}\left(\nabla \cdot \mathbf{E} - \rho/\varepsilon_0\right) = 0 \tag{2.12}$$

$$\Leftrightarrow \quad \nabla \cdot \frac{\partial}{\partial t}\mathbf{E} - \frac{\partial}{\partial t}\rho/\varepsilon_0 = 0 \tag{2.13}$$

$$\Leftrightarrow \quad c^2 \nabla \cdot (\nabla \times \mathbf{B} - \mu_0 \mathbf{j}) - \frac{\partial}{\partial t}\rho/\varepsilon_0 = 0 \tag{2.14}$$

$$\Leftrightarrow \quad \nabla \cdot \mathbf{j} + \frac{\partial}{\partial t}\rho = 0. \tag{2.15}$$

In other words, Gauss' law (2.5) is satisfied for all t, if (a) it is satisfied for $t = 0$ and (b) if the code rigorously respects conservation of charge. If one of these conditions is not met, the code would have to take care of Eqn. (2.5) by solving it at each time step in order to produce a physically correct electric field [Birdsall and Langdon, 1985; Buneman, 1993]. As the numerical integration of this equation is computationally expensive, it is highly desirable to meet conditions (a) and (b) in order to fulfill Eqn. (2.5) automatically.

Not having to solve Poisson's equation at each time step is of great advantage also in view of the later parallelization of the simulation. For the code to run efficiently in parallel, methods that update the fields purely from local data are preferred to "global" methods that require distant information and thus involve inter-processor communication. Leap-frogging $\mathbf{E}$ and $\mathbf{B}$ in the Yee grid via Eqns. (2.7) and (2.8) represents a local method that provides new data from old by using values of the immediate vicinity only. By contrast, solving Poisson's equation requires distant information: The potential anywhere depends on the electric charges everywhere and on spatial boundary conditions. Hence, if the code is designed to automatically satisfy Poisson's equation, the field update becomes fully local and can be parallelized in quite a straightforward manner.

The tasks one is left with in order to have a purely local field solver are (a) to satisfy Poisson's equation at $t = 0$ and (b) to make the code rigorously charge-conserving. While the former has to be dealt with as part of the initial conditions (Sec. 2.5), the latter affects the scatter routine, which distributes the current contribution of a moving particle among the grid points.

2.2 Scatter

In a particle simulation, charge ρ and current $\mathbf{j}$ arise from the distribution and the movement of the particles, respectively. In order to enter the field equations as source terms, ρ and $\mathbf{j}$ have to be known at the discrete grid points of the electromagnetic field grid. Since the particles can be located anywhere within the grid, their contributions to current and charge have to be extrapolated somehow from the particle position to the discrete grid points. This is done in the scatter routine.

2.2.1 Charge deposit

The obviously simplest method of extrapolation from a continuous particle coordinate to a discrete set of grid points is to assign the whole particle charge to the closest grid point (Fig. 2.5b). This weighting scheme is called nearest-grid-point method (NGP) or zero-order weighting and goes back to Burger [1964] and Hockney [1965]. NGP-weighting of a particle that moves with uniform velocity through the grid results in a charge that is jumping from grid point to grid point as the particle passes through the cell centres. Since this jumping introduces a large amount of noise, NGP is not used in today's simulations anymore.

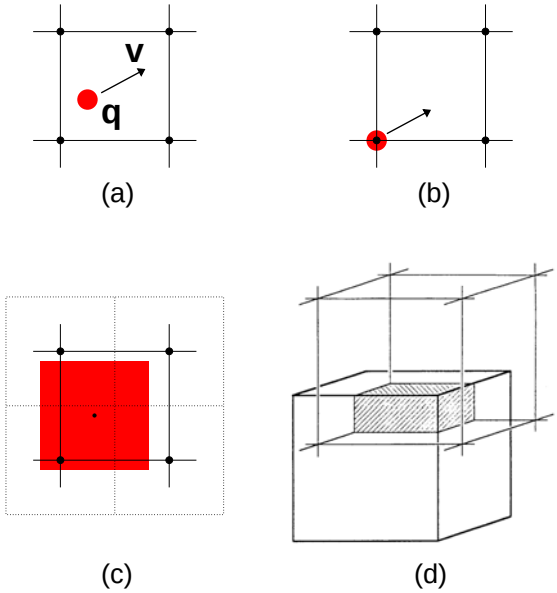

Figure 2.5: Different weighting schemes for a particle within a spatial grid (a): Nearest-grid-point (b), particle-in-cell/cloud-in-cell (c), PIC/CIC in 3D (d). The dotted lines in (c) represent the complementary grid.

A much smoother assignment scheme, which is probably the most commonly used, is the so-called particle-in-cell (PIC) or cloud-in-cell (CIC) method introduced by Harlow [1964]. As opposed to the point particles of the NGP method, the PIC scheme regards the particles as having a finite size that is equal to the size of the grid cell (Fig. 2.5c). The particle charge is considered to be evenly distributed over the particle surface (in 2D) or the particle volume (3D), and the fraction of the total particle charge that is assigned to each grid point is proportional to the intersection area (volume) of the particle with the respective complementary grid cell (dotted line in Fig. 2.5c). For obvious reasons, this scheme is also referred to as area weighting (volume weighting). In this way, a PIC-particle contributes to the charge at four (eight) different grid points in two (three) dimensions. On passing through the grid with constant velocity its deposited charges rise and fall in a linear fashion, rather than jump from grid point to grid point as with NGP.

Still smoother assignment schemes can be constructed by modifying the particle shape, i. e. the charge distribution within the particle. Splines are used to generate particle shapes with a gently outward decreasing charge density. Depending on the order of the splines being used, this leads to second and higher order weighting schemes [Hockney and Eastwood, 1981; Birdsall, 1991].

Choosing the appropriate weighting method means to make a trade-off between the desired reduction of numerical noise and the computational cost of the assignment scheme. In the wealth of plasma particle simulations that have been developed to date, the PIC method has proved to be a good compromise in this respect. For our simulation, we therefore make use of the PIC concept in that we use finite-sized particles with an evenly distributed charge. In order to further reduce noise we apply a smoothing routine to the obtained current distribution, which will be described further down.

2.2.2 Calculation of current

The current carried by a particle is proportional to the product of charge q and velocity $\mathbf{v}$. A particle's current contribution to a grid point is therefore commonly calculated by multiplying its charge as deposited by NGP, PIC or any other weighting method with the particle velocity. While such a computation of the current is very straightforward, it has the disadvantage of in general not satisfying the charge continuity equation

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \mathbf{j} = 0. \tag{2.16}$$

In other words, this method generates a current $\mathbf{j}$ whose divergence is not equal to the temporal change of the charge distribution by virtue of which it arose [Birdsall and Langdon, 1985; Marder, 1987]. As the rigorous conservation of charge was found to be indispensable for dropping the integration of Gauss' law (Sec. 2.1), we cannot rely on this simple method for the calculation of $\mathbf{j}$.

Several charge-conserving current calculation techniques have been proposed so far [Buneman and Pardo, 1968; Morse and Nielsen, 1971; Marder, 1987; Villasenor and Buneman, 1992], the most sophisticated and least noisy of which is the one of Villasenor and Buneman. These authors make use of the PIC method and define the current as normal vectors on the mid-points of the grid cell boundaries (Fig. 2.6). As each simulation step a moving particle sweeps over some of the cell boundaries with a certain subarea (subvolume) of its charge, the current on each boundary is calculated as the amount of charge that is carried through the boundary during the particle move. The movement of the particle from $\mathbf{x}$ to $\mathbf{x} + \Delta\mathbf{x}$ during Δt is thereby approximated as straight, irrespective of how the actual macroscopic particle movement looks like.

Fig. 2.6 shows a simple particle move from (x, y) to $(x + \Delta x, y + \Delta y)$ in a 2D grid. Following the scheme of Villasenor and Buneman, this would give rise to

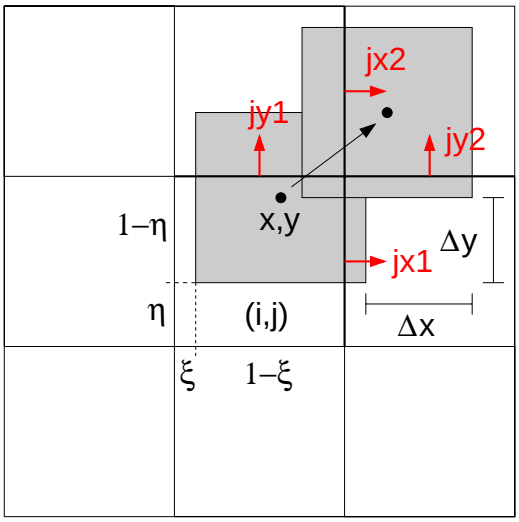

Figure 2.6: A particle move in a 2D grid: calculation of the currents according to the method of Villasenor and Buneman [1992].

currents through four cell faces: j_{x1}, j_{x2}, j_{y1}, and j_{y2}. In order to derive a compact formula for these currents, we introduce relative particle coordinates ξ and η for the position of the lower-left particle corner (x, y) within the grid cell (i, j) such that $\xi, \eta \in [0, 1)$:

$$x = X_i + (\xi + 0.5) \cdot \Delta X \tag{2.17}$$
$$y = Y_j + (\eta + 0.5) \cdot \Delta X\,, \tag{2.18}$$

where X_i and Y_j are the coordinates of the lower-left corner of the grid cell (i, j).

According to Fig. 2.6, the intersections of the particle with the four cell faces in question are $(1 - \eta) \cdot \Delta X$, $\eta \cdot \Delta X$, $(1 - \xi) \cdot \Delta X$, and $\xi \cdot \Delta X$, respectively. When the particle moves in a straight manner to its new position $(x + \Delta x, y + \Delta y)$, these intersections change linearly. For the computation of the particle area that is swept through each cell face during this move, we therefore have to consider the *averaged* intersections $\overline{(1 - \eta) \cdot \Delta X} = (1 - \overline{\eta}) \cdot \Delta X$ etc. Here $\overline{\xi}$ and $\overline{\eta}$ are the averages of the relative particle coordinates ξ and η between the respective values before and after the movement. With these definitions, the four currents of Fig. 2.6 can be computed as:

$$j_{x1} = \Delta x \cdot (1 - \overline{\eta}) \cdot Q/\Delta t \tag{2.19}$$
$$j_{x2} = \Delta x \cdot \overline{\eta} \cdot Q/\Delta t \tag{2.20}$$
$$j_{y1} = \Delta y \cdot (1 - \overline{\xi}) \cdot Q/\Delta t \tag{2.21}$$
$$j_{y2} = \Delta y \cdot \overline{\xi} \cdot Q/\Delta t, \tag{2.22}$$

where j_x and j_y have the dimension of a current density (i. e. current / length in 2D) and Q denotes the charge density of the particle (i. e. particle charge / ΔX^2).

In general, the set of cell faces that is affected by a particle move depends on the start and end position of the particle. Requirements for the numerical stability of the code constrain the extent of the particle movement during Δt to less than a grid constant (Sec. 2.7). This sets an upper limit for the number of cell faces that are swept over during a particle move. For 2D grids, Villasenor and Buneman found that apart from the 4-wall case of Fig. 2.6, also 7-wall and 10-wall sweeps are possible (Fig. 2.7). These cases are handled best by splitting the particle movement into two (7-wall case) or three parts (10-wall case) each of which can then be treated as a standard 4-wall sweep via Eqns. (2.19)-(2.22).

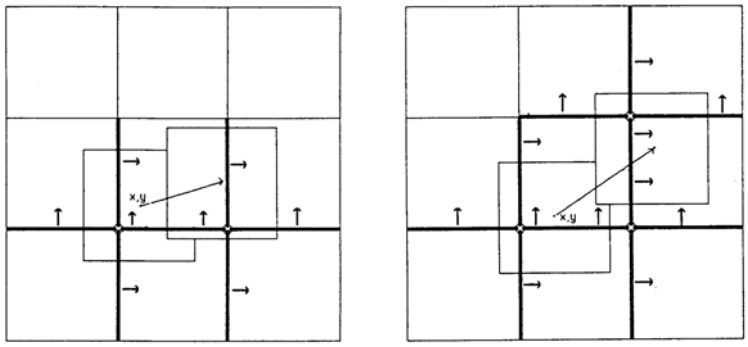

Figure 2.7: 7-wall and 10-wall moves of a particle in a 2D grid [modified after Villasenor and Buneman, 1992].

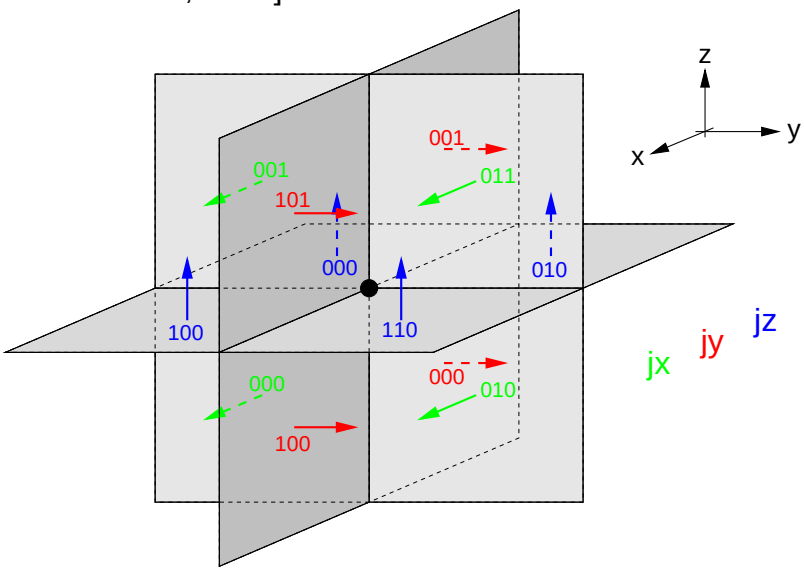

Figure 2.8: The 12 cell faces and associated currents for a particle move in a 3D grid. The indices refer to Eqns. (2.23)-(2.34).

In order to incorporate the current assignment scheme of Villasenor and Buneman into our simulation, it has to be generalized to three dimensions. To that end, we have to find the 3D "standard" particle move corresponding to the 4-wall case in the 2D grid, to which more complicated particle movements can be reduced by splitting them into several steps.

From Fig. 2.8 it becomes obvious that a particle of the size of one grid cell, which is *at rest* somewhere in the simulation domain, intersects a maximum of eight cells. For "small" movements from (x, y, z) to $(x + \Delta x, y + \Delta y, z + \Delta z)$, which do not cause the particle to leave this set of eight cells, it therefore carries charge through their 12 inner faces. For the current densities associated with such a movement, we obtained:

$$
\begin{aligned}
j_x(i,j,k) &= \left(\Delta x \cdot (1-\overline{\eta}) \cdot (1-\overline{\zeta}) + \chi\right) \cdot Q/\Delta t && (2.23)\\
j_x(i,j+1,k) &= \left(\Delta x \cdot \overline{\eta} \cdot (1-\overline{\zeta}) - \chi\right) \cdot Q/\Delta t && (2.24)\\
j_x(i,j,k+1) &= \left(\Delta x \cdot (1-\overline{\eta}) \cdot \overline{\zeta} - \chi\right) \cdot Q/\Delta t && (2.25)\\
j_x(i,j+1,k+1) &= \left(\Delta x \cdot \overline{\eta} \cdot \overline{\zeta} + \chi\right) \cdot Q/\Delta t && (2.26)
\end{aligned}
$$

$$
\begin{aligned}
j_y(i,j,k) &= \left(\Delta y \cdot (1-\overline{\zeta}) \cdot (1-\overline{\xi}) + \chi\right) \cdot Q/\Delta t && (2.27)\\
j_y(i,j,k+1) &= \left(\Delta y \cdot \overline{\zeta} \cdot (1-\overline{\xi}) - \chi\right) \cdot Q/\Delta t && (2.28)\\
j_y(i+1,j,k) &= \left(\Delta y \cdot (1-\overline{\zeta}) \cdot \overline{\xi} - \chi\right) \cdot Q/\Delta t && (2.29)\\
j_y(i+1,j,k+1) &= \left(\Delta y \cdot \overline{\zeta} \cdot \overline{\xi} + \chi\right) \cdot Q/\Delta t && (2.30)
\end{aligned}
$$

$$
\begin{aligned}
j_z(i,j,k) &= \left(\Delta z \cdot (1-\overline{\xi}) \cdot (1-\overline{\eta}) + \chi\right) \cdot Q/\Delta t && (2.31)\\
j_z(i+1,j,k) &= \left(\Delta z \cdot \overline{\xi} \cdot (1-\overline{\eta}) - \chi\right) \cdot Q/\Delta t && (2.32)\\
j_z(i,j+1,k) &= \left(\Delta z \cdot (1-\overline{\xi}) \cdot \overline{\eta} - \chi\right) \cdot Q/\Delta t && (2.33)\\
j_z(i+1,j+1,k) &= \left(\Delta z \cdot \overline{\zeta} \cdot \overline{\xi} + \chi\right) \cdot Q/\Delta t, && (2.34)
\end{aligned}
$$

where $\chi = \Delta x \Delta y \Delta z/12$. Analogously to ξ and η, we have introduced the relative particle coordinate ζ for the z component such that $z = Z_k + (\zeta + 0.5) \cdot \Delta X$. Q is now the "3D" charge density (particle charge / ΔX^3), and the current densities j have the dimension current / area.

In the case of more extended movements the particle might leave the set of eight grid cells of Fig. 2.8 in the x, y and / or z direction, each causing an intersection with another 8 cell faces. The maximum number of cell faces that are swept over by a particle during one time step is therefore 12+3×8=36. As suggested by Villasenor and Buneman [1992] for the 2D case, we handle a general particle move in our 3D code by splitting it into a maximum of four parts during each of which the particle can be assigned to an ensemble of eight grid cells similar to Fig. 2.8. Eqns. (2.23) to (2.34) with suitably chosen i, j and k can then be used to calculate the currents for each of these grid cell ensembles.

2.2.3 Smoothing

By making use of the PIC scheme with finite-sized rather than point-like particles, we already eliminate much of the unnatural noise created by coarse-graining the physically very fine-grained population of particles. As a further step towards

smoothing, we apply a straightforward form of filtering, namely averaging the current over neighbouring cells in all three dimensions: Rather than depositing the whole current as obtained by any of Eqns. (2.23)-(2.34) on grid cell (i, j, k), we follow the suggestion of Buneman, who employed this smoothing technique in his TRISTAN code [1993], and spread the current over the cube of $3 \times 3 \times 3$ grid cells surrounding (i, j, k). The weights to be used for the spreading of the current are those corresponding to a convolution of the current array with the series $\frac{1}{4} - \frac{1}{2} - \frac{1}{4}$ in all three dimensions. These are:

- 1/8 for the central cell (i, j, k) itself
- 1/16 for the six adjacent cells $(i \pm 1, j, k), (i, j \pm 1, k), (i, j, k \pm 1)$
- 1/32 for the twelve edge centres of the $3 \times 3 \times 3$ cube: $(i \pm 1, j \pm 1, k), (i \pm 1, j, k \pm 1), (i \pm 1, j \pm 1, k)$
- 1/64 for the eight cube corners $(i \pm 1, j \pm 1, k \pm 1)$.

Translated into Fourier space, this smoothing amounts to applying a low-pass filter to the current. Its effect in real space can be thought of as giving the particles a shape according to Fig. 2.9: cubes of $3 \times 3 \times 3$ grid cells with a charge distribution that decreases outward in a step-like fashion according to the weights above.

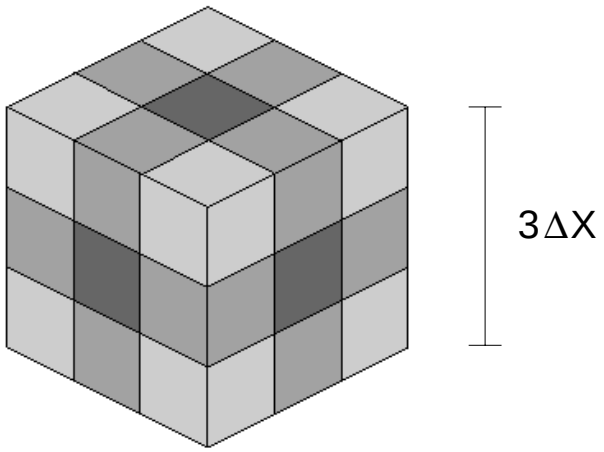

Figure 2.9: The effective particle shape according to the smoothing algorithm: a cube of $3 \times 3 \times 3$ grid cells with an outwardly decreasing charge density (darker grey tones correspond to higher charge densities).

The significant reduction of noise associated with smoothing has of course to be paid with a worsening of spatial resolution from one ΔX to something between one and three ΔX. However, a higher spatial resolution can always be recovered by decreasing the grid spacing ΔX, i. e. by increasing the overall grid size.

Through the smoothing procedure, the current calculation becomes computationally quite expensive: Each of the 12 current components obtained by Eqns. (2.23)-(2.34) has to be spread among 27 grid cells, which requires $12 \times 27 = 324$ current data accesses per particle for a single "12-wall movement". Considering that there are commonly much more than one particle in a grid cell, the great deal of work associated with smoothing calls for some optimization of the smoothing algorithm.

We therefore depart from Buneman's implementation of the smoothing procedure [1993], and do not calculate the smoothed current for *each* particle before summing it up to obtain the total current of all particles. We rather first compute the "original" (non-smoothed) current according to Eqns. (2.23)-(2.34) for *all* particles and then run through the whole grid to perform the smoothing by spreading the current according to the weights above. In other words, we carry out the smoothing on the grid (grid-oriented) while Buneman smoothes each particle contribution (particle-oriented).

For those grid cells with more than one particle inside, the grid-oriented realization of the smoothing algorithm is obviously more efficient than a particle-oriented smoothing. Typical particle densities in our simulation will range from 20 to 100 particles per grid cell within the plasma beam. For this region of the simulation domain, the grid-oriented smoothing results in a significant reduction of workload. Many of the grid cells farther away from the thruster beam can, however, be void of particles. In order to prevent the code from wasting time in spreading zeroes, we therefore first check if the current component to be spread is non-zero before applying the 27 smoothing steps.

Our grid-oriented implementation of Buneman's smoothing algorithm is advantageous also in view of the intended parallelization of the code: The achievable speed-up of a parallel code strongly depends on how evenly the total work is distributed among the different processors. As we will see later, it is much easier to balance the workload of a grid-oriented routine than of a routine that operates directly on the particles.

2.3 Gather

The gather procedure can be regarded as the inverse of the scatter procedure: While in the scatter step charge and current carried by the particles have to be *extrapolated* to the grid in order to serve as source terms for the field solve, the gather routine *interpolates* the fields from the grid onto the particles. It is important to note that every particle to which the fields are to be interpolated has itself contributed to these fields by virtue of its charge and current. Physically, the electric or magnetic field produced by a particle should not give any force back to it. In other words, particles should not feel a "self-force". As this has to be accounted for numerically, the gather step requires some caution.

Matsumoto and Omura [1985] investigated the nonphysical self-force, which can arise through an ill-coded gather routine, in detail and provided a recipe for its cancellation:

1. Before interpolating the field quantities to the particle, they have to be "re-located" to those grid points where the particle charge is defined.
2. The interpolation of the fields has to be performed with the same weighting scheme that is used for the scatter step.

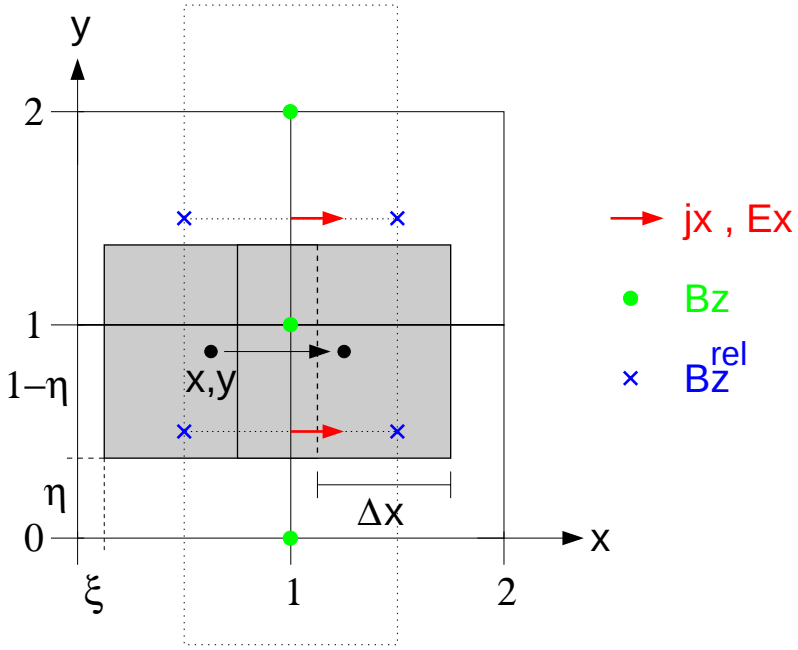

Figure 2.10: Calculation of magnetic self-force: a particle move in a 2D grid. The dotted lines represent the complementary grid.

How we implemented this recipe into our simulation and successfully eliminated the self-force is now illustrated for the magnetic field. We consider a single particle that moves by Δx within an initially field-free 2D Yee grid (Fig. 2.10). Eqns. (2.19)-(2.22) allow to compute the currents associated with this movement:

$$j_x(1,\tfrac{1}{2}) = \Delta x \cdot (1-\eta) \cdot Q/\Delta t \quad (2.35)$$

$$j_x(1,\tfrac{3}{2}) = \Delta x \cdot \eta \cdot Q/\Delta t\,. \quad (2.36)$$

According to Eqn. (2.8) these currents give rise to an electric field:

$$E_x(1,\tfrac{1}{2}) = -(1-\eta) \cdot Q\Delta x/\varepsilon_0 \quad (2.37)$$

$$E_x(1,\tfrac{3}{2}) = -\eta \cdot Q\Delta x/\varepsilon_0\,. \quad (2.38)$$

By virtue of the leap-frog algorithm, this electric field generates a magnetic field. In 2D, only the z component of the magnetic field exists. It is defined on the grid

cell corners and can be obtained according to Eqn. (2.7) as the negative circulation of $\mathbf{E}$ around each cell corner (dotted line in Fig. 2.10):

$$B_z(1,0) = -(1-\eta) = \eta - 1 \tag{2.39}$$
$$B_z(1,1) = -\big((-(1-\eta) - (-\eta)\big) = 1 - 2\eta \tag{2.40}$$
$$B_z(1,2) = -(-\eta) = \eta\,, \tag{2.41}$$

where, for simplicity, we have introduced the normalization $Q\Delta x\Delta t/(\varepsilon_0 \Delta X) := 1$. This is the magnetic field that is produced by the simple particle movement of Fig. 2.10. The Lorentz force $\mathbf{v} \times \mathbf{B}$ acting on the particle can now be computed as

$$\mathbf{v} \times \mathbf{B} = (v_x, 0, 0)^T (0, 0, B_p) = (0, -v_x B_p, 0)\,, \tag{2.42}$$

where B_p is the magnetic field at the particle as obtained by some interpolation scheme. While physically a self-generated magnetic field does not exert a net Lorentz force, it depends obviously on the choice of B_p to enter Eqn. (2.42), if this force also vanishes numerically.

In a naive approach, we might take e. g. the magnetic field value that is closest to the particle position, $B_z(1,1)$, as B_p in Eqn. (2.42) and would end up with an unphysical non-zero Lorentz force. This choice of B_p departs from the above recipe of Matsumoto and Omura in both ways: Neither are the magnetic field values relocated to those grid points where the charge is defined, i. e. the cell centres, nor does it employ the area weighting scheme, with which the currents were previously deposited on the grid. In order to obtain the magnetic field at the particle position, it rather uses a simple nearest-grid-point weighting.

The correct gather routine, which we constructed by applying the procedure suggested by Matsumoto and Omura [1985, from now on referred to as "the M-O-recipe"] to our simulation and which successfully eliminates the nonphysical self-force, works as follows: First, the magnetic field values have to be relocated to where the charge is defined. Since within our current calculation scheme, the current is computed via the charge that is sweeping through cell *boundaries* when moving from one cell to another, the locations where the charge itself is defined are obviously the cell *centres*. Therefore, the relocated $\mathbf{B}$-fields are also defined in the cell centres. They are obtained by averaging the fields of the four cell corners:

$$B_z^{rel}\left(\tfrac{1}{2}, \tfrac{1}{2}\right) = \tfrac{1}{4}\big(B_z(0,0) + B_z(1,0) + B_z(0,1) + B_z(1,1)\big) = -\eta/4 \tag{2.43}$$
$$B_z^{rel}\left(\tfrac{3}{2}, \tfrac{1}{2}\right) = \tfrac{1}{4}\big(B_z(1,0) + B_z(2,0) + B_z(1,1) + B_z(2,1)\big) = -\eta/4 \tag{2.44}$$
$$B_z^{rel}\left(\tfrac{1}{2}, \tfrac{3}{2}\right) = \tfrac{1}{4}\big(B_z(0,1) + B_z(1,1) + B_z(0,2) + B_z(1,2)\big) = (1-\eta)/4 \tag{2.45}$$
$$B_z^{rel}\left(\tfrac{3}{2}, \tfrac{3}{2}\right) = \tfrac{1}{4}\big(B_z(1,1) + B_z(2,1) + B_z(1,2) + B_z(2,2)\big) = (1-\eta)/4\,. \tag{2.46}$$

The relocated magnetic field values B_z^{rel} now have to be interpolated ("gathered") to the particle position. According to the M-O-recipe, this interpolation has to employ the same weighting scheme that was previously used to deposit the currents,

i. e. area weighting in the 2D case of Fig. 2.10. Hence, the respective weights for the four cells are

$$w_1 = (1-\xi)\cdot(1-\eta) \tag{2.47}$$
$$w_2 = \xi\cdot(1-\eta) \tag{2.48}$$
$$w_3 = (1-\xi)\cdot\eta \tag{2.49}$$
$$w_4 = \xi\cdot\eta\,, \tag{2.50}$$

and the magnetic field at the particle position is calculated as:

$$\begin{aligned} B_p &= w_1\cdot B_z^{rel}\left(\tfrac{1}{2}\,,\tfrac{1}{2}\right) + w_2\cdot B_z^{rel}\left(\tfrac{3}{2}\,,\tfrac{1}{2}\right) + w_3\cdot B_z^{rel}\left(\tfrac{1}{2}\,,\tfrac{3}{2}\right) + w_4\cdot B_z^{rel}\left(\tfrac{3}{2}\,,\tfrac{3}{2}\right) \\ &= \big((1-\xi)+\xi\big)\cdot(1-\eta)\cdot(-\eta/4) + \big((1-\xi)+\xi\big)\cdot\eta\cdot(1-\eta)/4 \\ &= (1-\eta)\cdot(-\eta/4) + \eta\cdot(1-\eta)/4 \\ &= 0\,. \end{aligned} \tag{2.51}$$

The interpolated magnetic field vanishes indeed, and the particle is not subjected to a magnetic self-force. It is by virtue of relocating the fields to where they arose from and of using the same weighting scheme as in the scatter step, that the gather routine acquires a symmetry that cancels out the particle's own contributions to the magnetic field.

In three dimensions, the gather procedure works in a similar manner. The relocation to the cell centres involves averaging over eight cell corners, and the respective interpolation weights for the eight cells of Fig. 2.8 are

$$w(i,j,k) = (1-\xi)\cdot(1-\eta)\cdot(1-\zeta) \tag{2.52}$$
$$w(i+1,j,k) = \xi\cdot(1-\eta)\cdot(1-\zeta) \tag{2.53}$$
$$w(i,j+1,k) = (1-\xi)\cdot\eta\cdot(1-\zeta) \tag{2.54}$$
$$w(i+1,j+1,k) = \xi\cdot\eta\cdot(1-\zeta) \tag{2.55}$$
$$w(i,j,k+1) = (1-\xi)\cdot(1-\eta)\cdot\zeta \tag{2.56}$$
$$w(i+1,j,k+1) = \xi\cdot(1-\eta)\cdot\zeta \tag{2.57}$$
$$w(i,j+1,k+1) = (1-\xi)\cdot\eta\cdot\zeta \tag{2.58}$$
$$w(i+1,j+1,k+1) = \xi\cdot\eta\cdot\zeta\,. \tag{2.59}$$

While our 3D implementation of the M-O-recipe successfully cancels the self-force of the magnetic field, we still have to check if it also eliminates the electric self-force. To prove that analytically in three dimensions involves some tedious algebra. We therefore restrict ourselves to one dimension, and leave the verification of the actual 3D case to numerical testing.

Particles in a 1D grid can be thought of as infinite planes of thickness ΔX. We consider such a plane-particle at rest in the 1D grid of Fig. 2.11. At $t = n$ let the electric force on this particle E_p^n be zero. Using the above described interpolation procedure it can be calculated as follows:

$$\begin{aligned} E_p^n &= w_1^n\cdot E_x^{rel,n}\left(\tfrac{1}{2}\right) + w_2^n\cdot E_x^{rel,n}\left(\tfrac{3}{2}\right) \\ &= (1-\xi)\cdot(E_{x0}^n + E_{x1}^n) + \xi\cdot(E_{x1}^n + E_{x2}^n) \stackrel{!}{=} 0\,. \end{aligned} \tag{2.60}$$

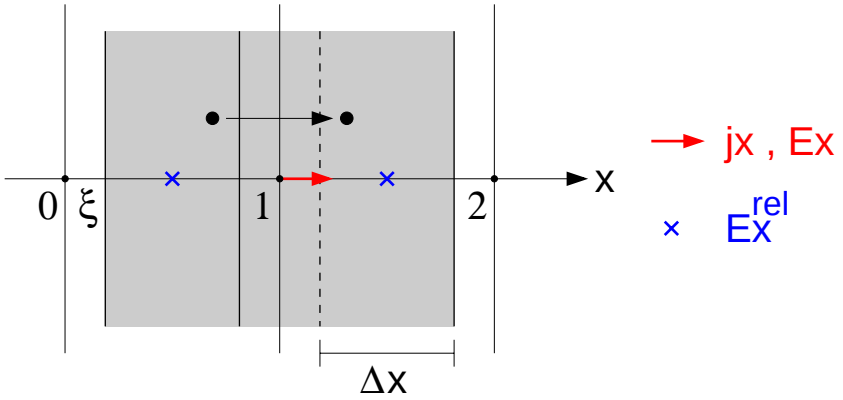

Figure 2.11: Calculation of electric self-force: a particle move in a 1D grid.

Let the particle now move from x to $x + \Delta x$. According to Eqn. (2.19), this gives rise to a current

$$j_{x1} = \Delta x \cdot Q/\Delta t\,, \tag{2.61}$$

where Q has the dimension of charge / length. This current enters the electric field update Eqn. (2.8) to give

$$\begin{aligned} E_{x0}^{n+1} &= E_{x0}^{n} && (2.62)\\ E_{x1}^{n+1} &= E_{x1}^{n} - \Delta x \cdot Q/\varepsilon_0 && (2.63)\\ E_{x2}^{n+1} &= E_{x2}^{n}\,. && (2.64) \end{aligned}$$

Hence, the relocated fields for $t = n + 1$ are

$$\begin{aligned} E_x^{rel}\left(\tfrac{1}{2}\right) &= \tfrac{1}{2}\left(E_{x0}^{n+1} + E_{x1}^{n+1}\right) = \tfrac{1}{2}\left(E_{x0}^{n} + E_{x1}^{n} - \Delta x \cdot Q/\varepsilon_0\right) && (2.65)\\ E_x^{rel}\left(\tfrac{3}{2}\right) &= \tfrac{1}{2}\left(E_{x1}^{n+1} + E_{x2}^{n+1}\right) = \tfrac{1}{2}\left(E_{x1}^{n} - \Delta x \cdot Q/\varepsilon_0 + E_{x2}^{n}\right), && (2.66) \end{aligned}$$

and the interpolated field at the particle position, E_p^{n+1}, becomes

$$\begin{aligned} E_p^{n+1} &= \tfrac{1}{2}\left[(1-\xi-\Delta\xi)\cdot E_x^{rel}\left(\tfrac{1}{2}\right) + (\xi+\Delta\xi)\cdot E_x^{rel}\left(\tfrac{3}{2}\right)\right]\\ &= \tfrac{1}{2}\cdot\Delta x\cdot\left[\left(E_{x2}^{n+1} - E_{x0}^{n+1}\right)/\Delta X - Q/\varepsilon_0\right]\,, \end{aligned} \tag{2.67}$$

where we have introduced the normalized displacement $\Delta\xi = \Delta x/\Delta X$. The right hand side of Eqn. (2.67) is the self-generated electric field as "felt" by the particle. Physically, this field has to vanish. If this is the case also numerically depends on the term in square brackets

$$\left(E_{x2}^{n+1} - E_{x0}^{n+1}\right)/\Delta X - Q/\varepsilon_0\,, \tag{2.68}$$

which is identified as Gauss' law $\nabla\cdot\mathbf{E} = \rho/\varepsilon_0$ in 1D, integrated from $x = 0$ to $x = 2$ in Fig. 2.11. In other words, our implementation of the gather routine does not generate an electric self-force, if and only if the code satisfies Gauss' law. From our discussions in Sec. 2.1 we saw that by virtue of the field update of Eqns. (2.7)-(2.8) and of the charge-conserving current assignment scheme, Gauss' law is automatically satisfied. Hence, our gather routine does neither generate a magnetic nor an electric self-force. That this also holds in 3D was successfully verified by computing the self-consistent electromagnetic forces for random displacements of single particles in vacuum.

2.4 Particle push

In the particle push routine, the equations of motion are integrated in order to update particle positions and velocities. The forces acting on a particle arise from the electromagnetic field and, possibly, from collisions with other particles. While $\mathbf{E}$ and $\mathbf{B}$ at the locations of the particles have already been made available by the gather routine, we still have to concern ourselves with particle collisions.

Possible particle collisions in the thruster's Xe^+-e-plasma are electron-ion, electron-electron and ion-ion collisions. The respective collision frequencies ν can be estimated as

$$\nu_{ei} \approx \omega_{pe}^0 / N_D \tag{2.69}$$

$$\nu_{ee} \approx \nu_{ei} \tag{2.70}$$

$$\nu_{ii} = \left(\frac{m_e}{m_i}\right)^{\frac{1}{2}} \left(\frac{T_e}{T_i}\right)^{\frac{3}{2}} \nu_{ee}\,, \tag{2.71}$$

where N_D is the number of electrons in a Debye sphere [Chen, 1974, p. 352]. Substituting the DS1 values of Table 1 and assuming $T_i = 0.04 T_e$ [Wang et al., 1996] results in

$$\nu_{ei} \approx \nu_{ee} \approx 2.5 \cdot 10^4 \text{ s}^{-1} \text{ and } \nu_{ii} \approx 6 \cdot 10^3 \text{ s}^{-1}\,. \tag{2.72}$$

Electron-ion and electron-electron collisions are the most probable. Rather than their absolute values, it is, however, the ratio between collision frequencies and characteristic frequencies of the plasma, such as the electron plasma frequency ω_{pe}, that determines the importance of collisions for our simulations. According to Eqn. (2.69) this ratio is given by the number of particles in a Debye sphere, which is $N_D \approx 1.5 \times 10^5$ for the case of DS1. Hence, after 100 electron plasma periods, which is the typical time interval covered by our simulations, only every 1500th particle has collided. On the *time* scale of our simulation we can therefore regard the plasma as collisionless. The *spatial* scale on which this is justified can be estimated as the electron and ion mean free paths, which are

$$\lambda_e = v_{e0}^{th} / \nu_{ee} \approx 24 \text{ m} \tag{2.73}$$

$$\lambda_i = v_{i0} / \nu_{ei} \approx 1.4 \text{ m}\,. \tag{2.74}$$

As long as our simulation does not significantly exceed these scales, we do not have to consider a collision term in the equations of motion. These are thus

$$\frac{d}{dt}(\gamma m_s \mathbf{v}) = q_s \left(\mathbf{E} + \mathbf{v} \times \mathbf{B}\right) \tag{2.75}$$

$$\frac{d\mathbf{x}}{dt} = \mathbf{v}\,, \tag{2.76}$$

where m_s and q_s are particle mass and charge of plasma species s. Strictly speaking, rather than m_s and q_s, the mass m_S and charge q_S of the *super-particles* should appear in Eqn. (2.75). However, since the super-particles consist of a certain number of real plasma particles, they have the same charge to mass ratio as the real particles of the corresponding plasma species: $q_S/m_S = q_s/m_s$. The equations of

motion are therefore just the same. As our simulation is an electromagnetic code, where velocities as high as the velocity of light can occur, we include relativistic effects in Eqn. (2.75) via $\gamma = 1/\sqrt{1 - v^2/c^2}$.

For the integration of the equations of motion, we employ a standard algorithm developed by Boris [1970], which leap-frogs $\mathbf{x}$ and $\mathbf{u} := \gamma\mathbf{v}$ in a time-centred fashion as follows:

$$\mathbf{u}^{n+1/2} = \mathbf{u}^{n-1/2} + \frac{q_s \Delta t}{m_s}\left(\mathbf{E}^n + \frac{\mathbf{u}^{n+1/2} + \mathbf{u}^{n-1/2}}{2\gamma^n} \times \mathbf{B}^n\right) \tag{2.77}$$

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \frac{\mathbf{u}^{n+1/2}}{\gamma^{n+1/2}}\Delta t \ , \tag{2.78}$$

with γ being obtained via $\gamma = \sqrt{1 + u^2/c^2}$. Position $\mathbf{x}$ and velocity $\mathbf{u}$ are defined at full and half integer steps, respectively. For the velocity update with Eqn. (2.77), both $\mathbf{E}$ and $\mathbf{B}$ are needed at full integer steps. While the electric field is indeed provided at full integers by the field solver described in Sec. 2.1, the magnetic field is available only at half integers. Therefore, the magnetic field update Eqn. (2.7) has to be split into two steps, each of which advances $\mathbf{B}$ only half a time step. In this way, the magnetic field is defined at both full and half integer time steps (cf. Fig. 2.12).

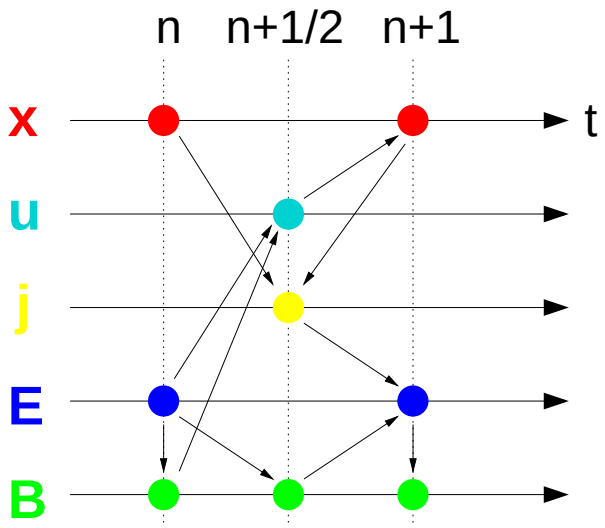

Figure 2.12: The definitions of position, velocity, current, electric and magnetic field in time and their interdependencies. Note that the magnetic field is defined at both full and half integer steps.

To compute the Lorentz force $\mathbf{u} \times \mathbf{B}$ in Eqn. (2.77), also the velocity is needed at $t = n$. As this is not known, Boris' method approximates it as the average between old and new velocity $(\mathbf{u}^{n+1/2} + \mathbf{u}^{n-1/2})/2$. To obtain $\mathbf{u}^{n+1/2}$ from the now implicit Eqn. (2.77), Boris suggested the following procedure:

1. Half electric acceleration: $\mathbf{u}^- = \mathbf{u}^{n-1/2} + \frac{q_s}{2m_s}\mathbf{E}^n \Delta t$

2. Magnetic rotation (Fig. 2.13): $\mathbf{u}^+ - \mathbf{u}^- = (\mathbf{u}^+ + \mathbf{u}^-) \times \mathbf{b_0}$

   with $\mathbf{b_0} := q_s \mathbf{B}^n \Delta t / 2\gamma^n m_s$

3. Another half electric acceleration: $\mathbf{u}^{n+1/2} = \mathbf{u}^+ + \frac{q_s}{2m_s}\mathbf{E}^n \Delta t.$

This step-wise integration of Eqn. (2.77) allows to implement the effect of the magnetic field with high accuracy as a real rotation of the vector $\mathbf{u}^-$. While above this integration step is given in implicit form to illustrate its rotational character (cf. Fig. 2.13), the simulation code computes $\mathbf{u}^+$ explicitly [Boris, 1970; Birdsall and Langdon, 1985; Buneman, 1993] via

$$\mathbf{u}^+ = \mathbf{u}^- + 2\,\frac{\mathbf{u}^- + \mathbf{u}^- \times \mathbf{b_0}}{1 + b_0^2} \times \mathbf{b_0}\,. \tag{2.79}$$

From Fig. 2.13 it can be seen that the total angle of rotation in the plane perpendicular to $\mathbf{B}$ is

$$\Theta = 2 \arctan |\mathbf{b_0}| = 2 \arctan \left(\frac{q_s}{2\gamma^n m_s} \mathbf{B}^n \Delta t \right). \tag{2.80}$$

As $\arctan \vartheta \approx \vartheta$ for "small" ϑ, this amounts to

$$\Theta \approx \frac{q_s}{\gamma^n m_s} \mathbf{B}^n \Delta t =: \Omega_s \Delta t\,, \tag{2.81}$$

which is the exact angle of rotation for a real gyrating particle during Δt. The error, i. e. $|\Theta - \Omega_s \Delta t|$, is about 3% for a time step Δt of one tenth of a gyroperiod and less than 1% for Δt being a twentieth of a gyroperiod.

Among the physical scenarios to which we will apply our simulation are some with practically "infinitely" strong magnetic fields. We therefore have to check if the particle push produces sensible results for such fields. As the particle velocity component along $\mathbf{B}$ is not affected by the magnetic rotation of Eqn. (2.79), it will continue to behave correctly even for infinite $\mathbf{B}$. Concerning the perpendicular components, however, Eqn. (2.80) yields a rotation angle of $\Theta = \pi$ for $\mathbf{B} \to \infty$. Hence, very strong magnetic fields result in a reversal of the perpendicular particle velocities after each time step. This particle behaviour corresponds

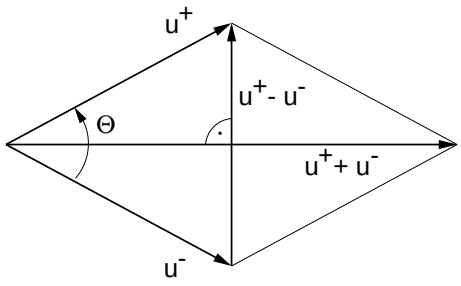

Figure 2.13: The second step of Boris' particle push procedure: the magnetic rotation of u^- to obtain u^+. Shown are the projections into the plane perpendicular to b_0 [after Birdsall and Langdon, 1985].

to the actual effect of an infinitely strong magnetic field in that it inhibits a net movement across $\mathbf{B}$. However, rather than a circular gyromotion the particles perform a linear movement with a spatial extent of $l = v_\perp \Delta t$, where $v_\perp$ is the perpendicular velocity. For simulations with very strong magnetic fields, it is this length scale l that has to be taken as the effective "gyroradius". The analytical gyroradius, which tends to zero for $\mathbf{B} \to \infty$, is of no numerical relevance in that case.

2.5 Initial and boundary conditions

Among many other thinkable physical problems, our simulation code is designed to be applicable to both of the ion thruster-induced plasma regimes that were stated in the introduction: the neutralization and the solar wind interaction. Each simulation scenario requires quite individual initial and boundary conditions. As the scope of this work is the investigation of the neutralization process, the initial and boundary conditions to be presented in the following refer to this scenario.

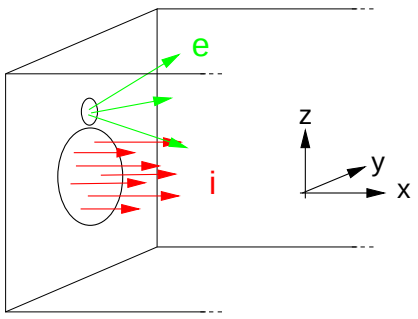

Figure 2.14: The standard geometry for simulations of ion thruster beam neutralization. Electrons are injected through the small top opening, ions through the bigger one in the bottom.

The general geometric configuration for simulating ion thruster beam neutralization looks like in Fig. 2.14. Electrons and ions are injected from one side into the 3D simulation box. Apart from these two plasma species originating from the thruster itself, the actual environment of an ion thruster as described in Sec. 1 also comprises charge exchange ions (CEX) and the ambient solar wind. The densities of these constituents are in the order of 10^{12} m^{-3} for the CEX ions around DS1 [Wang et al., 2000] and 10^{6} m^{-3} for the solar wind at 1 AU, which is by factors of 4×10^{3} and 4×10^{9} smaller than the DS1 ion beam density (Table 1).

Hence, for the neutralization process, i. e. the mixing between electrons and ions ejected by the ion thruster, the CEX ions and the ambient solar wind plasma are of no importance. When modelling the neutralization scenario they can be completely disregarded. Therefore, the initial condition for our simulations of ion thruster beam neutralization is the simplest one can think of: a field-free vacuum. Somewhat more complicated are, however, the boundary conditions.

2.5.1 Boundary conditions: Fields

Any field solve method has to rely on boundary conditions in order to obtain the electric and / or magnetic field throughout the simulation domain. These conditions should be chosen to match the actual physical situation of the simulated scenario. For electrostatic simulations, a sensible choice might be to set $\mathbf{E} \equiv 0$ along the boundary, implying that the boundaries are situated a long way from the region of computational interest, where the fields are already decayed. In terms of electromagnetics, however, $\mathbf{E} \equiv 0$ would correspond to a perfectly conducting boundary. Electromagnetic waves incident upon such a boundary would be reflected back into the computational domain and would build up standing waves. This behaviour deviates dramatically from the actual physical scenario that we want to simulate. The boundaries in our simulation should rather act like an infinite region of free space, letting out all waves that are generated within the simulation volume.

In one dimension, such an “open” boundary can be constructed quite elegantly [Birdsall and Langdon, 1985]: Maxwell’s equations are equivalent to the wave equation

$$\left(\frac{\partial^2}{\partial t^2} - c^2 \frac{\partial^2}{\partial x^2} \right) \begin{pmatrix} \mathbf{E} \\ \mathbf{B} \end{pmatrix} = 0 \,, \tag{2.82}$$

which has the well-known solutions

$$\mathbf{E}, \mathbf{B} \propto \exp\left(i\omega t \pm ikx\right) \quad \text{with} \quad \omega = kc \,, \tag{2.83}$$

where the plus sign corresponds to a wave propagating in $-x$ direction and the minus sign to a wave propagating in $+x$ direction. By factorizing the wave oper-

ator $\partial_t^2 - c^2\partial_x^2$, Eqn. (2.82) can be cast into two equations

$$\left(\frac{\partial}{\partial t} - c\frac{\partial}{\partial x}\right)\begin{pmatrix}\mathbf{E}\\ \mathbf{B}\end{pmatrix} = 0 \tag{2.84}$$

$$\left(\frac{\partial}{\partial t} + c\frac{\partial}{\partial x}\right)\begin{pmatrix}\mathbf{E}\\ \mathbf{B}\end{pmatrix} = 0\,, \tag{2.85}$$

each of which is satisfied by *either* left-going waves (2.84) *or* right-going waves (2.85). The realization of an open boundary on e. g. the left side of the simulation domain consists now of using Eqn. (2.84) to advance the boundary values of $\mathbf{E}$ and $\mathbf{B}$ in time:

$$\frac{\partial}{\partial t}\begin{pmatrix}\mathbf{E}\\ \mathbf{B}\end{pmatrix} = c \cdot \frac{\partial}{\partial x}\begin{pmatrix}\mathbf{E}\\ \mathbf{B}\end{pmatrix}. \tag{2.86}$$

As right-going waves do not satisfy Eqn. (2.84), this ensures the suppression of reflected waves. Similarly, Eqn. (2.85) is employed to update $\mathbf{E}$ and $\mathbf{B}$ at the right boundary, thus suppressing left-going waves there.

Lindman [1975] extended these 1D open boundaries to two and three dimensions. By decomposing any wave coming from within the computational region into a superposition of plane waves and projecting these waves with a suitably chosen projection operator to the boundary normal, he is able to play back the multi-dimensional problem to the 1D case above. His boundary conditions can handle propagating and evanescent waves incident at almost any angle and proved to be very efficient for electromagnetic waves in vaccum and in plasmas [Buneman et al., 1992, Buneman, 1993; Wang et al., 1995].

We implemented Lindman's boundaries into our simulation code. However, they turned out to be incompatible with our simulation scenario: Test runs showed that in the presence of strong electrostatic fields, the boundary values of the electric field as obtained by Lindman's scheme diverge. Since strong electrostatic fields close to the boundary are inherent to our simulation with its spatially separated electron and ion sources, Lindman's boundaries are useless for our purposes.

Rather than trying to modify Lindman's elegant and powerful albeit very sensitive boundaries according to our needs, we decided to rely on a much more robust method for suppressing the reflection of outgoing electromagnetic waves: *absorption* in an electrically conducting boundary layer [e. g. Tajima and Lee, 1981].

An electromagnetic wave entering an electrically conducting region generates dissipative Ohmic currents. It is spatially damped on the scale of the skin length

$$l_s(\omega, \sigma) = \sqrt{\frac{2}{\mu_0 \omega \sigma}}\,, \tag{2.87}$$

where σ is the electric conductivity (Fig. 2.15a). Numerically, a finite conductivity can be realized by running an additional loop over the electric field in the

conducting region that modifies it according to

$$\mathbf{E} := \mathbf{E} - \mathbf{j_c}\,\Delta t/\varepsilon_0 = (1 - \sigma\Delta t/\varepsilon_0)\cdot\mathbf{E}\,, \tag{2.88}$$

which corresponds to adding an Ohmic current $\mathbf{j}_c = \sigma\mathbf{E}$ to the particle current $\mathbf{j}$ that enters the $\mathbf{E}$-field update via Eqn. (2.8).

The magnetic field and the normal component of the electric field can be determined self-consistently via the common leap-frog algorithm, if the tangential components of the electric field are prescribed at the outer edge of the conductive layer. Setting them to zero, as we do in our simulation, results in a perfect reflection there. Thus, waves impinging on a conducting boundary are damped on the way in, reflected and further damped on the way out. The effective damping length is therefore $2d$, when d is the thickness of the conducting layer (Fig. 2.15b).

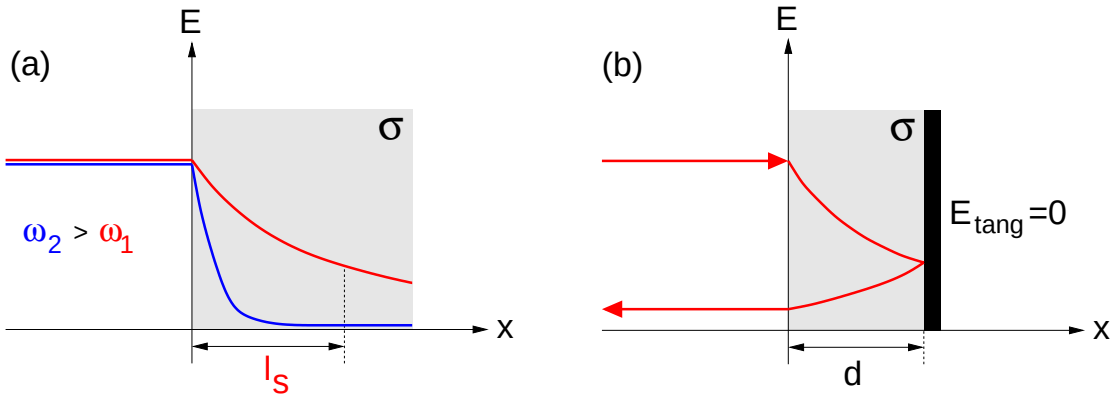

Figure 2.15: Conducting wall boundaries: (a) electromagnetic wave damping in an electrically conducting medium (l_s: skin length), (b) numerical realisation of a conducting boundary of thickness d.

The residual wave amplitude upon returning into the computational domain depends according to Eqn. (2.87) on wave frequency and conductivity. For a given frequency, higher conductivities result obviously in smaller residual amplitudes, i. e. in a better absorption of the wave. However, a certain fraction of the energy carried by the wave is reflected already at the inner interface between simulation volume and conducting region. This fraction becomes larger for increasing conductivity contrasts. Hence, the total absorption of the wave cannot be enhanced by simply increasing σ.

In our simulations, we fix the boundary thickness at $d = 5\Delta X$ and choose σ such that the effective damping length $2d$ equals the skin length of that frequency where most wave power is to be expected, i. e. the electron plasma frequency:

$$2d \stackrel{!}{=} l_s(\omega = \omega_{pe}) \Rightarrow \sigma = \frac{1}{2\mu_0\omega_{pe}d^2}\,. \tag{2.89}$$

With this choice of σ, the reflection coefficient for electromagnetic wave energy can be reduced to less than 0.1. Even though not being as efficient as Lindman's, our boundary conditions are very robust and, in particular, compatible with strong electrostatic fields. Moreover, as we will see in the following, a conductive boundary layer is also a sensible choice in terms of particle boundary conditions, as it facilitates the proper removal of leaving particles.

2.5.2 Boundary conditions: Particles

Electrons and ions are continuously injected into the simulation volume through one side and are removed once they reach one of the six simulation boundaries. If there are N particles to be injected per time step, then we actually inject *one* particle every $\Delta t/N$ in order to smooth the injection process in time. We therefore supply each new particle with its individual injection time $t_{inj} = 0, \ldots, (N-1) \cdot \Delta t/N$ and advance it once with a special push routine that takes account of the individual t_{inj}.

Hence, the particle boundary conditions for our simulations of ion thruster beam neutralization are, in principle, very simple. More caution is required in order to make the injection of new and the removal of old particles consistent with the field solving algorithm.

A characteristic of charge-conserving local field solvers is that the charge density ρ does not appear explicitly, so that particles give rise to electric fields not via their charges but only by virtue of the currents they produce. Imagine a particle of charge $+q$ "injected" into vacuum at $\mathbf{x}_0$ with a velocity $\mathbf{v}$, that moves to $\mathbf{x}_1 = \mathbf{x}_0 + \mathbf{v}\Delta t$ after one time step and is forced to stay there (Fig. 2.16). As a consequence of charge conservation, the electric field that will develop from such a configuration is not a monopole Coulomb-field corresponding to the positive charge at $\mathbf{x}_1$, but a dipole field caused by $+q$ at $\mathbf{x}_1$ and a virtual negative charge $-q$ at $\mathbf{x}_0$. The emergence of such virtual charges of opposite sign, or in other words of artificial divergences of $\mathbf{E}$, in charge-conserving codes may be troublesome in certain cases and has to be dealt with.

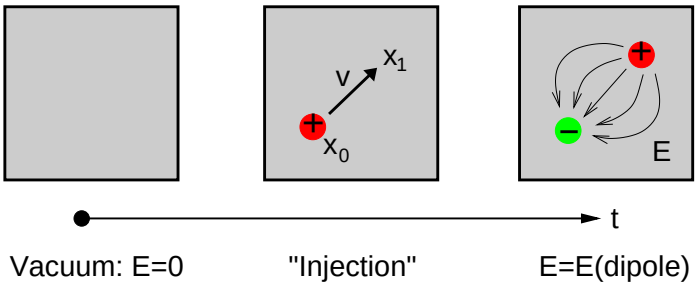

Figure 2.16: The emergence of virtual charges upon particle injection.

One way of injecting a particle without generating a virtual charge of opposite sign would be to add the analytically known electrostatic monopole field of the injected particle to each grid point of the simulation volume. This would alter ρ and $\mathbf{E}$ in a self-consistent manner. However, such a procedure is non-local, computationally very expensive and might cause stability problems as it implicitly involves an infinite propagation velocity of the electric field.

Another cure for unwanted divergences of $\mathbf{E}$ is to generate particles in pairs of oppositely signed charges that are injected at the same location $\mathbf{x}_0$ [see also Buneman, 1993]. Since this amounts to injecting "neutral" particles, no correction to the electric field has to be made. This method is local and easy to implement, also on parallel computers. However, due to the creation of charges in pairs, it can only be applied as long as the plasma to be injected is quasi-neutral. It does not work in situations that require the injection of non-neutral plasmas, such as the simulation of ion thrusters with their spatially separated electron and ion sources. For this kind of geometries, the proper choice of the injection method is essential, as will be illustrated in the following.

Using our 3D electromagnetic PIC code with the charge-conserving current assignment method as desribed in Sec. 2.2.2, we have simulated various injection schemes for a simplified ion thruster configuration. The results of our simulation runs are shown in Fig. 2.17. Randomly distributed over the top box on the left side of the simulation volume, electrons are created each time step and are injected with a bulk velocity of $v_x = 0.1c$ plus an isotropic thermal spread of $0.01c$. With the same number per time step, ions are created in the lower box and leave it with a uniform bulk velocity of $0.1c$ plus a negligibly small thermal spread corresponding to $T_e = T_i$. In order to make the ions practically insensitive to electromagnetic fields, their mass was chosen to be as high as 250,000m_e. We switched on the injection at $t = 0$, stopped it after a while and let the system evolve for some time. While the densities (top panels in Fig. 2.17) were obtained by summing over the particles, the divergences of $\mathbf{E}$ (bottom panels) were calculated by central-differencing the electric field.

Fig. 2.17a shows the results for the case of straightforward injection without any correction to the electric field. As can be seen from the top panel, an injected ion "drop" has left the injection box and has flown up to $x \approx 7$. In accordance with Maxwell's equations, the electric field shows a positive divergence in this region (bottom panel). A divergence of similar amplitude but of opposite sign has emerged within the rectangle from where the ions were injected. This is a direct consequence of the charge-conserving character of the code, and is exactly what was described above as the creation of oppositely signed virtual charges.

The virtual charges exert an attractive force on the injected particles. While the huge ion mass prevents the ion dynamics from being significantly altered by this artificial force, the effects on the electrons are dramatic. Similar to the emergence of negative divergences of $\mathbf{E}$ in the ion source region, the electron injection in the top rectangle gives rise to positive virtual charges there. As a consequence,

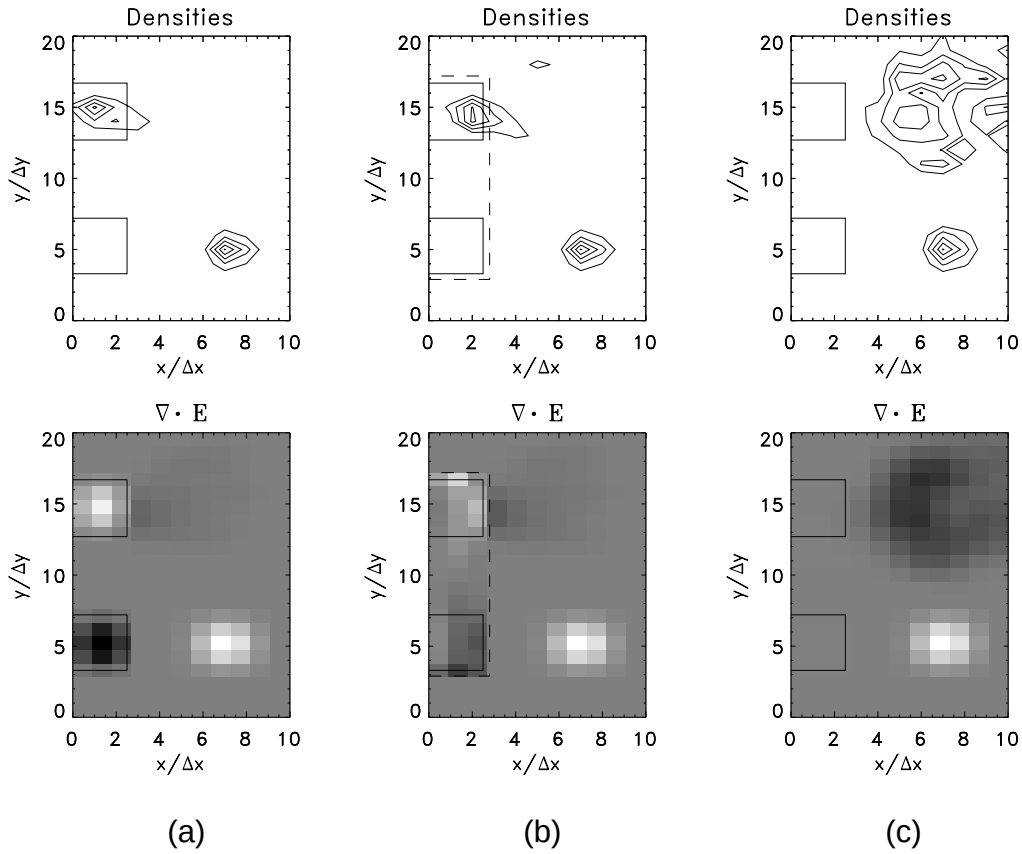

Figure 2.17: Cuts through the simulation domain including the electron and ion injection regions (rectangles adjacent to the left border). *Top panels:* electron and ion densities. *Bottom panels:* divergence of the electric field. Black and white correspond to negative and positive divergence, respectively. The background grey tone indicates zero divergence. (a) No compensation, (b) with a conductive region within the dashed rectangle, (c) using the "generator technique". Note the divergencelessness of the electric field in the injection regions and the free expansion of the electrons for case (c).

the relatively light electrons cannot expand freely and are stuck in the injection region, despite having the same bulk velocity as the ions.

As each injected particle creates one virtual charge, a continuous injection of plasma particles would lead to an ever increasing amount of artificial charges in the particle source regions. Their electric field would rise correspondingly, so that from a certain point onwards the whole simulation dynamics would be ruled by the artificial charges. Hence, such an injection scheme without any compensation for the emergence of virtual charges forbids itself.

In the simulated configuration, positive and negative virtual charges emerge at the same rate. Their continuous accumulation could therefore be suppressed if

they were allowed to compensate each other. The easiest way of doing that is to conductively connect both source regions of virtual charges by embedding them in an electrically conductive medium. Such a conductive connection would allow the electric fields of the virtual charges to drive currents that, in turn, contribute to the removal of the artificial divergences of $\mathbf{E}$. To that end, we embedded the electron and ion injection boxes into a region of finite electrical conductivity, which is marked by the dashed lines in Fig. 2.17b. As described in Sec. 2.5.1, the conductivity in this region is modelled by modifying the electric field according to

$$\mathbf{E} := (1 - \sigma \Delta t / \varepsilon_0) \cdot \mathbf{E} \,. \tag{2.90}$$

We simulated the same scenario as described above for various values of $\sigma' := \sigma \Delta t / \varepsilon_0$. A conductivity around $\sigma' = 0.05$ turned out to be a reasonable choice and lead to results like the one depicted in Fig. 2.17b. As can be seen from the bottom panel, the conductivity indeed provides a certain degree of compensation of virtual charges, when compared to Fig. 2.17a. However, along the surface of the conductive region, the divergences of $\mathbf{E}$ are still quite strong and apparently big enough to inhibit most of the electrons from escaping. These surface divergences could not be significantly reduced by other choices of σ. Hence, in case of such a locally concentrated massively non-neutral injection of particles as needed for the simulation of ion thrusters, a conductive region does not serve as a cure for virtual charges.

In order to completely remove the artificial divergences of $\mathbf{E}$, a rigorous mutual compensation between the virtual charges of both particle source regions has to be enforced. We accomplished this by implementing a kind of electrical generator into the injection process, which generates the particles *before* they are injected: As in many plasma simulation applications, in our example electrons and ions are injected with the same number per time step and can therefore be grouped into electron-ion pairs. To each of these pairs belongs a set of positions $\mathbf{x}_e, \mathbf{x}_i$ and velocities $\mathbf{v}_e, \mathbf{v}_i$. The role of the generator now is to place these particles not at their respective injection positions, but halfway between $\mathbf{x}_e$ and $\mathbf{x}_i$ (see Fig. 2.18), and to provide them with individually determined transfer velocities $\mathbf{v}_{tr}$ and $-\mathbf{v}_{tr}$ that make them reach their respective injection locations $\mathbf{x}_e$ and $\mathbf{x}_i$ in a certain number of time steps. During their transfer, the particles are not subjected to electromagnetic fields. However, the currents they produce are collected and enter Eqn. (2.8). Once the particles reach their destinations $\mathbf{x}_e$ and $\mathbf{x}_i$, they are provided with their original injection velocities $\mathbf{v}_e$ and $\mathbf{v}_i$ and appear as new particles in the simulation.

As electrons and ions are created in pairs at the same location, this injection scheme rigorously satisfies conservation of charge and thus guarantees that no unwanted divergences of $\mathbf{E}$ develop. Fig. 2.17c shows that employing the "generator technique" has indeed dramatic consequences for the injection scenario, when compared to the previous injection schemes. As expected from a proper particle injection, the electric field in the particle source regions is divergence-free and the electrons can expand freely.

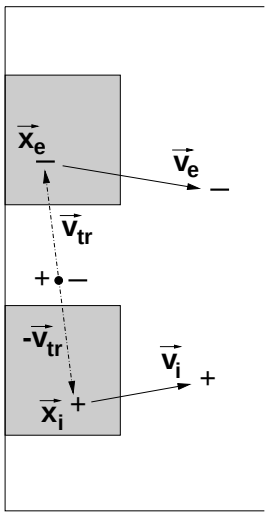

Figure 2.18: The "generator": New electrons and ions are placed in pairs halfway between their respective injection positions and are moved to $\mathbf{x}_e$ and $\mathbf{x}_i$. During this transfer, they are not subjected to the ambient electromagnetic field, but the currents they produce contribute already to the field update of Eqn. (2.8).

The generator requires additional storage and computing time for the particles that are in the transfer phase. When N is the number of electron-ion pairs that are injected each time step, and T_{tr} is the number of time steps that are needed for the transfer, then this additional workload amounts to $2NT_{tr}$ particles. N is in general determined by the actual simulation project, so that the only way of reducing storage and computing time is to choose T_{tr} as small as possible. A short duration of the transfer phase T_{tr} can be achieved by a high transfer velocitiy v_{tr}, whose upper limit is of course the velocity of light c. Allowing v_{tr} to be close to c thus reduces storage requirements and workload. However, this has to be paid with enhanced emission of radiation: On the transition between the transfer phase and the actual injection of the particles, their velocities change abruptly from $\mathbf{v}_{tr}$ to $\mathbf{v}_{e/i}$, possibly involving a 90° direction change as in our sample simulation. The amount of radiation generated via this massive particle acceleration increases with v_{tr} getting closer to c, which might be troublesome.

We carried out a series of simulation runs of our sample configuration with varying transfer velocities between $0.1c$ and $0.8c$ and employed our conducting wall boundaries. No major changes in the particle dynamics were experienced. Hence, when appropriate absorbing boundary conditions such as ours are used, the enhanced radiation of the generator does not seem to be a problem.

Hence, with our "generator" that creates the charges in accordance with charge conservation *before* they are injected, we have developed a method that prevents the emergence of unwanted divergences of $\mathbf{E}$ in local electromagnetic field solvers. It allows a self-consistent injection of non-neutral plasmas as needed for configurations with spatially separated electron and ion sources.

After having been injected by the generator, a particle eventually reaches one of the computational boundaries, where it has to be removed from the simulation. Simply striking it off the roll of active particles is, however, not enough: By virtue

of the charge-conserving field solver, its *charge* remains at the particle's ultimate position, as if the particle got stuck in the boundary. Therefore, every particle that "leaves" the simulation volume gives rise to a stationary Coulomb field.

However, as we have chosen electrically conducting boundaries (Sec. 2.5.1), the charges of stuck particles are blurred and charges of opposite sign can compensate each other. That this weakens the single particle Coulomb fields efficiently has already been verified by Buneman [1993], who implemented a thin conducting layer adjacent to the simulation boundaries into his TRISTAN code.

2.6 Normalization

The foregoing sections have introduced the difference equations that are solved in our simulation code. They involved many multiplications with ΔX, Δt or q_s/m_s. These time consuming operations can be removed by a suitable normalization of the involved physical quantities. The normalization we use in our simulation is based on the one suggested by Matsumoto and Omura [1985] and looks as follows:

Distance	$\mathbf{x}^*$	=	$\frac{1}{\Delta X}\,\mathbf{x}$
Time	t^*	=	$\frac{2}{\Delta t}\,t$
Velocity	$\mathbf{v}^*$	=	$\frac{\Delta t}{2\Delta X}\,\mathbf{v}$
Charge to mass ratio	$\left(\frac{q}{m}\right)_s^*$	=	$\frac{m_e}{\lvert e\rvert}\frac{q_s}{m_s}$
Electric field	$\mathbf{E}^*$	=	$\frac{\lvert e\rvert}{m_e}\left(\frac{\Delta t}{2}\right)^2\frac{1}{\Delta X}\,\mathbf{E}$
Magnetic field	$\mathbf{B}^*$	=	$\frac{\lvert e\rvert}{m_e}\frac{\Delta t}{2}\,\mathbf{B}$
Current density	$\mathbf{j}^*$	=	$\frac{1}{\varepsilon_0}\frac{\lvert e\rvert}{m_e}\left(\frac{\Delta t}{2}\right)^3\frac{1}{\Delta X}\,\mathbf{j}$
Nabla operator	∇^*	=	$\Delta X\,\nabla$

The normalized quantities (marked with an asterisk) are dimensionless. They lead to a much simpler form of the various difference equations. One of these,

namely the calculation of the current from the particle movement, deserves further attention, because it controls to a certain degree the consistency of simulating real plasma particles as an ensemble of super-particles.

The normalized current density is defined as

$$\mathbf{j}^* = \frac{1}{\varepsilon_0}\frac{|e|}{m_e}\left(\frac{\Delta t}{2}\right)^3 \frac{1}{\Delta X}\,\mathbf{j}\,, \tag{2.91}$$

where $\mathbf{j}$ is the non-normalized current density, whose components are calculated according to Eqns. (2.23)-(2.34) as

$$j = Q_S \Delta V^* \frac{\Delta X}{\Delta t}\,. \tag{2.92}$$

Here, $Q_S = q_S/\Delta X^3$ is the charge density of the super-particle of species s and ΔV^* is the dimensionless fraction of the particle volume that is swept through the cell face. Substituting this into Eqn. (2.91) yields for the components of the normalized current

$$j^* = \frac{1}{\varepsilon_0}\frac{|e|}{m_e}\frac{\Delta t^2}{8} Q_S \Delta V^*\,. \tag{2.93}$$

In order for j^* to enter the $\mathbf{E}$-field update, the charge density of the super-particle Q_S has to be specified. It is obtained by requiring that the plasma frequency of the super-particles ω_{pS} be equal to the "real" plasma frequency of the corresponding species ω_{ps}:

$$\omega_{ps}^2 \stackrel{!}{=} \omega_{pS}^2 \tag{2.94}$$

$$\Leftrightarrow\quad \omega_{ps}^2 = \frac{\left(Q_S \Delta X^3\right)^2 N_S}{\varepsilon_0 \Delta X^3 m_S}\,, \tag{2.95}$$

where N_S is the super-particle density, i. e. the number of super-particles per grid cell. This equality ensures the consistency between the real plasma and its numerical representation via super-particles in terms of their oscillatory behaviour. The determination of Q_S is therefore a crucial step.

Solving Eqn. (2.95) for Q_S and substituting it into Eqn. (2.93) now allows to compute j^* via the following expression:

$$j^* = \frac{\left(\omega_{ps}\Delta t\right)^2}{N_S}\left(\frac{m}{q}\right)_s^* \Delta V^*\,. \tag{2.96}$$

While the particle volume fraction ΔV^*, which is carried through the various cell faces during the particle move, is determined for each cell face via Eqns. (2.23)-(2.34), the quantities N_S, $(q/m)_s$ and $\omega_{ps}\Delta t$ belong to or can be derived from the input parameters to be discussed in the following.

2.7 Numerical stability and input parameters

The overall numerical stability of the code requires to fulfill certain criteria. First of all, as we use an explicit field solving algorithm, time step Δt and cell size ΔX have to meet the Courant condition, which reads in 3D

$$\Delta t < \frac{1}{\sqrt{3}} \cdot \frac{\Delta X}{|v|}\,, \tag{2.97}$$

where v is the highest physical velocity in the simulation [e. g. Birdsall and Langdon, 1985]. Being an electromagnetic code, our simulation has to fulfill the Courant condition with v as the velocity of light. For a given cell size ΔX this results in very small time steps Δt. In order to cover the physical processes of interest with a numerically reasonable simulation length of 10^4–$10^5 \Delta t$, the involved physical process have to be "accelerated" by rescaling characteristic velocities of the system towards higher values. Alternatively, one can use an artificial velocity of light slower than the real one, to bring the time scales closer together. Either "trick" constitutes a deviation from the real physical system. However, there are two reasons why this does not present a major limitation to the code's applicability:

1. Rather than by absolute velocity values, the physics of ion thruster neutralization will turn out to be determined by *ratios* of characteristic velocities.

2. For the simulated plasma scenario, the velocity of light has no other physical significance than being the propagation speed for changes in the electromagnetic field. Using an artificially low velocity of light does therefore not change the plasma physics qualitatively, as long as the particles still perceive the fields as propagated practically instantaneously [Buneman, 1993]. How low the velocity of light can be chosen, depends on the individual simulation scenario. For example, Buneman et al. [1992] and Nishikawa [1997, 1998] reduced it to just twice the maximum particle velocity in their simulations of the Earth's magnetosphere.

Another stability criterion concerns the cell size ΔX itself and applies to simulations of "warm" plasmas, i. e. plasmas with a thermal electron velocity comparable to other characteristic system velocities. Okuda [1972] showed that for cell sizes greater than about three Debye lengths, a numerical instability develops that heats the plasma unphysically until $\Delta X \approx 3\lambda_D$. Details of the instability mechanism can be found in Okuda's original paper and in Birdsall and Langdon [1985].

When simulating "cold" plasmas with negligible thermal velocities, the Debye length gets close to zero, and there is no such restriction on ΔX. It can greatly exceed λ_D without causing instabilities [Birdsall and Langdon, 1985]. However, as in our case the electron thermal velocity is one of the fundamental velocities

of the simulated physical system and not at all negligible, we have to respect Okuda's condition on the cell size ΔX.

As far as characteristic frequencies of the plasma, such as gyrofrequency and plasma frequency are concerned, the mere stability of the particle push algorithm requires the respective periods to be greater than $\pi\Delta t$. For an inaccuracy of less than 1%, they have to be longer than $20\Delta t$.

In order to match all these stability criteria, the input parameters of the simulation have to be chosen accordingly. They are contained in a parameter file that is read upon program start (Fig. 2.19). As most of the input parameters are self-explanatory, we only point out the fundamental ones:

- The spatial resolution r_x, which determines the grid cell size via $\Delta X = r_x \lambda_D$. According to Okuda's criterion, r_x must not be bigger than around 3.
- The temporal resolution r_t representing the deviation of the used time step from Courant's maximum time step (cf. Eqn. 2.97):
$$\Delta t = \Delta t_{max}/r_t = \frac{1}{\sqrt{3}\, r_t} \cdot \frac{\Delta X}{c} \,.$$
Its theoretical minimum value is $r_t = 1$, which results in the field solve algorithm being marginally stable. In practice, it is chosen greater than 1.15.
- The electron thermal velocity upon injection v_{e0}^{th} normalized to the velocity of light c, where we understand the thermal velocity as $v_{e0}^{th} = \sqrt{2k_B T_{e0}/m_e}$.
- The frequency ratio between electron gyrofrequency and electron plasma frequency Ω_e/ω_{pe}.

These four parameters already suffice to completely determine the simulated plasma, i. e. they allow to compute the characteristic plasma time and length scales in numerical units Δt and ΔX. For instance, the electron plasma frequency can be obtained by noting that

$$\omega_{pe} = \frac{1}{\sqrt{2}} \frac{v_{e0}^{th}}{\lambda_D} \tag{2.99}$$

and using the above relations for λ_D and Δt:

$$\omega_{pe}\Delta t = \frac{1}{\sqrt{2}} \frac{v_{e0}^{th}}{\lambda_D} \Delta t = \frac{1}{\sqrt{2}}\, v_{e0}^{th} \frac{r_x}{\Delta X} \Delta t = \frac{1}{\sqrt{6}} \frac{v_{e0}^{th}}{c} \frac{r_x}{r_t} \,. \tag{2.100}$$

Similarly, the electron inertia length l_e in units of ΔX can be computed via

$$l_e = \frac{c}{\omega_{pe}} = \frac{\sqrt{2}}{r_x} \frac{c}{v_{e0}^{th}} \Delta X \,, \tag{2.101}$$

which only involves input parameters that are provided in the parameter file.

```
------------Number of processors in x
10
------------Number of processors in y
1
------------Number of processors in z
1
------------Number of species (including electrons)
2
------------Super-particle density in the beam (N_S)
20
------------Number of grid-points in x-direction
350
------------Number of grid-points in y-direction
80
------------Number of grid-points in z-direction
80
------------Number of time steps
7000
------------Time step for diagnostics
20
------------r_x (dx=r_x*l_Debye)
1.D0
------------r_t (dt=dt_max/r_t)
1.5D0
------------Conductivity of absorbing boundaries (sigma')
0.05D0
------------Thickness of conductivity layer
5
------------Electron gyro period / electron plasma period
0.D0
------------Electron thermal velocity / c
0.1D0
------------Electron beam velocity / c
0.D0
------------Electron beam width (in grid cells)
10
------------Beam spacing (in grid cells)
0
------------SPECIES2: charge/|e|
1.D0
------------SPECIES2: mass/m_e
250000.D0
------------SPECIES2: number density/electron density
1.D0
------------SPECIES2: temperature/electron temperature
1.D0
------------SPECIES2: beam velocity / c
0.025D0
------------SPECIES2: beam width (in grid cells)
25
------------End of inppar
```

Figure 2.19: The input parameter file *inppar*.

When inspecting Eqn. (2.100) for $\omega_{pe}\Delta t$, it seems that the plasma frequency is determined by the electron thermal velocity, which is quite contra-intuitive. However, rather than ω_{pe} itself, it is the time step Δt that depends on v_{e0}^{th}: For a given ω_{pe}, the electron thermal velocity determines the Debye length according to Eqn. (2.99), i. e. the cell size, which in turn controls the time step Δt via the Courant condition. A similar argumentation applies to the electron inertia length l_e (Eqn. 2.101). It is not l_e itself, but the cell size ΔX that depends on the electron thermal velocity.

Above we discussed two options for bringing the respective time scales of the electromagnetic fields and the plasma itself closer together: reducing the velocity of light or increasing the characteristic plasma velocities. As can be seen from Fig. 2.19, all the velocities provided as input quantities are normalized to the velocity of light. In this formulation, we are able to cover both of these options. Hence, it is left to the interpretation of the simulation results if one regards the numerical velocity of light to be downscaled or the plasma velocities to be upscaled. In any case, the rescaling of numerical quantities to physical values has to be done by matching ratios of characteristic lengths, times or velocities [e. g. Wang et al., 1996].

2.8 Selected tests

All the necessary ingredients of our simulation code have been presented in the preceding sections. In order to make sure that they produce correct physics they have to be tested. As it is quite a straightforward task to check the functioning of each of the different routines separately, such as particle push or field solve, we will restrict ourselves in the following to illustrate some selected verifications of their correct interplay.

2.8.1 Dispersion relation

We simulate an elongated grid of 64×5×5 cells with boundaries that are now *periodic* in all three dimensions both for fields and for particles. It is uniformly filled with a Maxwellian plasma consisting of electrons and practically immobile ions ($m_i = 10^8 m_e$). The electron thermal velocity is $v_{e0}^{th}/c = 10^{-3}$, and the spatial resolution is set to $\Delta X = 10^3 \lambda_D$, i. e. we simulate a dense plasma with negligible λ_D. The other input parameters are chosen such that the electron plasma frequency is $\omega_{pe}\Delta t = 0.22\pi$.

The finite thermal velocity of the electrons gives rise to thermal noise, which shows up in the electromagnetic field. The electrostatic and electromagnetic waves constituting this noise follow the dispersion characteristics of the plasma. The

noise spectrum of the electric field should therefore reproduce the plasma dispersion relation.

Due to the periodic boundaries, the simulation volume acts like a plasma-filled resonator. The wave field consists of eigenmodes with wavelengths matching the cell dimensions. For simplicity, we are interested in the one-dimensional dispersion relation, i. e. $\omega = \omega(k_x)$. This is obtained by stapling the electric field in the y and z direction, which cancels out the wave modes with k_y and k_z other than zero, and subsequent Fourier transformation.

Fig. 2.20 shows the spectral densities for the three components of the electric field in the first Brillouin zone, i. e. within $0 \leq k_x \Delta X \leq \pi$ (the upper edge of this zone, $k_x \Delta X = \pi$, is the analogue to the well-known Nyquist frequency of signal theory). The theoretical dispersion relations – the black curves in Fig. 2.20 – are

$$\omega_{lg}(k_x) = \omega_{pe} \tag{2.102}$$

for the longitudinal component E_x and

$$\omega_{tr}(k_x) = \sqrt{\omega_{pe}^2 + c^2 k_x^2} \tag{2.103}$$

for the transverse components E_y and E_z.

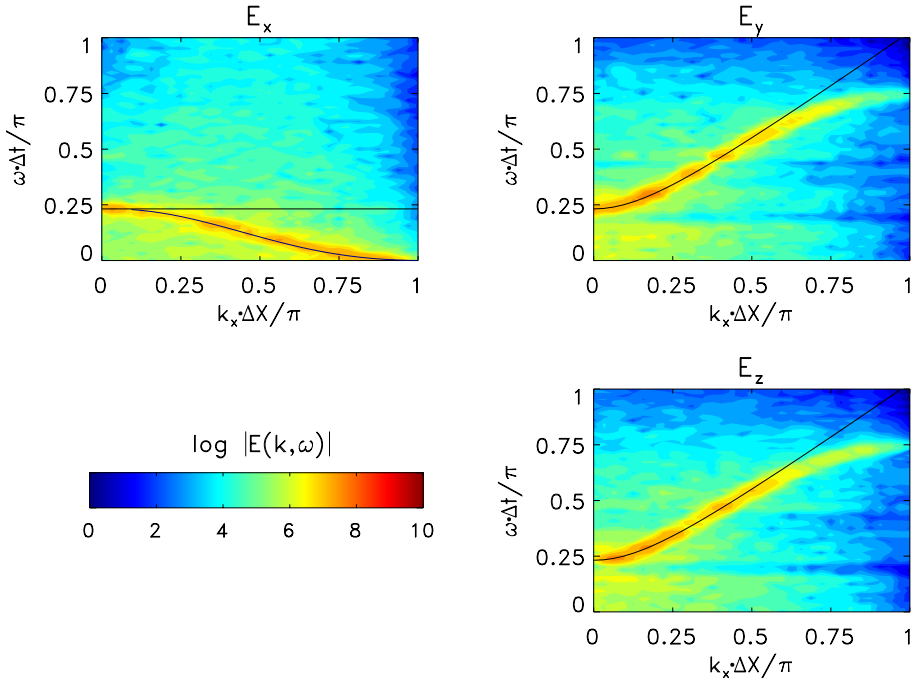

Figure 2.20: Spectral densities of the longitudinal and transverse electric field components in the first Brillouin zone (logarithmic scale, arbitrary units). The black lines represent the theoretically expected dispersion relation.

Simulation and theory correspond very well for the transverse components in the region below $k_x \Delta X < \pi/2$, including the cutoff at ω_{pe}. For larger k_x, however, the dispersion branch as obtained by the simulation bends down and exhibits a horizontal tangent upon reaching the edge of the Brillouin zone. This is an inherent feature of spatially discretized wave equations and is a well-known phenomenon not only of numerical simulations [Birdsall and Langdon, 1985], but also of some natural "discrete" systems such as phonons in a crystal lattice [Kittel, 1999]. Hence, the cell size of the simulation grid should be chosen small enough so that wavelengths of physically important waves extend at least over $4\Delta X$.

The spectral power below ω_{pe} arises from wave modes with $k_y, k_z \neq 0$ that did not cancel out perfectly by the stapling in y and z. They do not belong to the 1D dispersion relation and do therefore not constitute a deviation from theory.

The dispersion of the longitudinal component E_x, however, departs significantly from the theoretically expected course. Rather than staying at ω_{pe}, independent of k_x, it varies according to

$$\omega = \omega_{pe} \frac{\sin^2(k_x \Delta X)}{(k_x \Delta X)^2} = \omega_{pe} \operatorname{sinc}^2(k_x \Delta X) \,, \tag{2.104}$$

with the sinc-function $\operatorname{sinc}\theta = \sin\theta/\theta$. This behaviour is a consequence of simulating the plasma as an ensemble of *finite-sized* particles, as will be illustrated in the following.

A real plasma can be considered as consisting of a finite number of point-like particles. The density n of a plasma species can therefore be written as a sum of delta-functions:

$$n(x) = \sum_p \delta(x - x_p) \,, \tag{2.105}$$

where the sum goes over all particles p. The charge density ρ_p is readily obtained by

$$\rho_p(x) = q n(x) = \sum_p q \delta(x - x_p) \tag{2.106}$$

with q being the particle charge.

If now the point-like particles are replaced by finite-sized super-particles, their charge density ρ_S can be computed as

$$\rho_S(x) = \sum_p q S(x - x_p) \,, \tag{2.107}$$

where S is the so-called "shape factor" [Birdsall and Langdon, 1985]. It describes the distribution of charge within the super-particle. However, rather than directly representing this distribution, $S(x_0)$ determines which fraction of a super-particle located at $x = 0$ would be assigned to a fictitious grid point at $x = x_0$. It is therefore not to be confused with the actual shape of the particle. Fig. 2.21 shows the

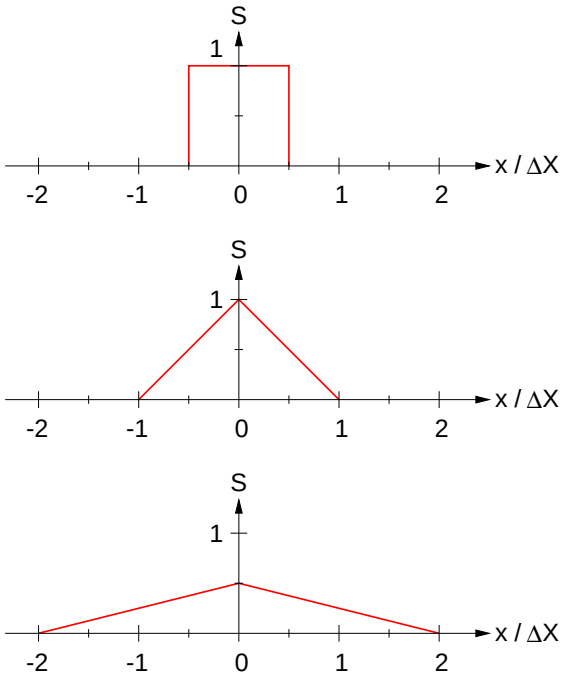

Figure 2.21: The shape factors of different weighting schemes: NGP (top panel), PIC (middle), and PIC combined with smoothing according to Sec. 2.2.3 (bottom panel) [first two panels modified after Birdsall, 1991].

shape factors for NGP- and PIC-weighting as well as for the weighting scheme that is applied in our code: PIC plus subsequent smoothing via a convolution with $\frac{1}{4} - \frac{1}{2} - \frac{1}{4}$ (cf. Sec. 2.2.3).

By writing the shape factor as

$$S(x - x_p) = \int S(x - x')\,\delta(x' - x_p)\,dx' \tag{2.108}$$

we obtain for the super-particle charge density

$$\begin{aligned} \rho_S(x) &= \sum_p q \int S(x - x')\,\delta(x' - x_p)\,dx' \\ &= \int S(x - x') \left(\sum_p q\delta(x' - x_p) \right) dx' \,, \end{aligned} \tag{2.109}$$

where we identify the sum over p as the point-particle charge density ρ_p at x' (Eqn. 2.106). Hence, the super-particle charge density ρ_S is the convolution of the shape factor S with the point-particle charge density ρ_p:

$$\begin{aligned} \rho_S(x) &= \int S(x - x')\,\rho_p(x')\,dx' \\ &= (S * \rho_p)\,(x) \,. \end{aligned} \tag{2.110}$$

In Fourier space this takes on the simple form

$$\rho_S(k) = S(k)\rho_p(k)\,, \tag{2.111}$$

where $S(k)$ is the Fourier-transformed of the shape factor $S(x)$. Hence, composing the plasma of finite-sized particles introduces a k-dependence of the charge density and thus of the plasma frequency!

Considering that the plasma frequency can be written as

$$\omega_{pe} = \sqrt{\frac{ne^2}{\varepsilon_0 m_e}} = \sqrt{\frac{\rho^2}{n\varepsilon_0 m_e}}\,, \tag{2.112}$$

its k-dependence in the simulation can be computed with Eqn. (2.111) as

$$\omega_{pe}^S(k) = \sqrt{\frac{\rho_S^2}{n\varepsilon_0 m_e}} = \sqrt{\frac{S^2(k)\rho_p^2}{n\varepsilon_0 m_e}} = |S(k)|\,\omega_{pe}\,. \tag{2.113}$$

For the weighting schemes displayed in Fig. 2.21, the Fourier-transformed of the shape factors are

$$S(k) = \mathrm{sinc}\,(k\Delta X/2) \quad \text{for NGP,} \tag{2.114}$$
$$S(k) = \mathrm{sinc}^2(k\Delta X/2) \quad \text{for PIC, and} \tag{2.115}$$
$$S(k) = \mathrm{sinc}^2(k\Delta X) \quad \text{for our weighting scheme (Fig. 2.22).} \tag{2.116}$$

In our simulation, the numerical dispersion relation for the longitudinal component E_x should therefore be

$$\omega(k) = \omega_{pe}^S(k) = \omega_{pe}\,\mathrm{sinc}^2(k_x\Delta X)\,, \tag{2.117}$$

in perfect consistency with the observations (Fig. 2.20 and Eqn. 2.104).

The unphysical longitudinal dispersion of short wavelengths imposes a restriction on the size of the super-particles and thus on the grid cell spacing. According to Fig. 2.20, it should be chosen such that $k\Delta X \lesssim \pi/3$, i. e. $\lambda \gtrsim 6\Delta X$ for physically important waves.

We note that the artificial k-dependence of the charge density arises already via the use of finite-sized particles; it is not an effect of the spatial grid. The spatial discretization of the field quantities, however, has further consequences for the dispersion relation, the most important of which is wave aliasing: Density perturbations with wavenumbers greater than $\pi/\Delta X$ are folded back as aliases into the first Brillouin zone. This is a well-known phenomenon in discrete time series analysis and is usually dealt with by appropriate filtering. In a particle simulation model, the shape factor $S(k)$ plays the role of a filter (cf. Eqn. 2.111). However, in contrast to time series analysis, wavenumbers beyond the resolution do not only reappear as aliases in the spectrum, but – by virtue of the gather-scatter process –

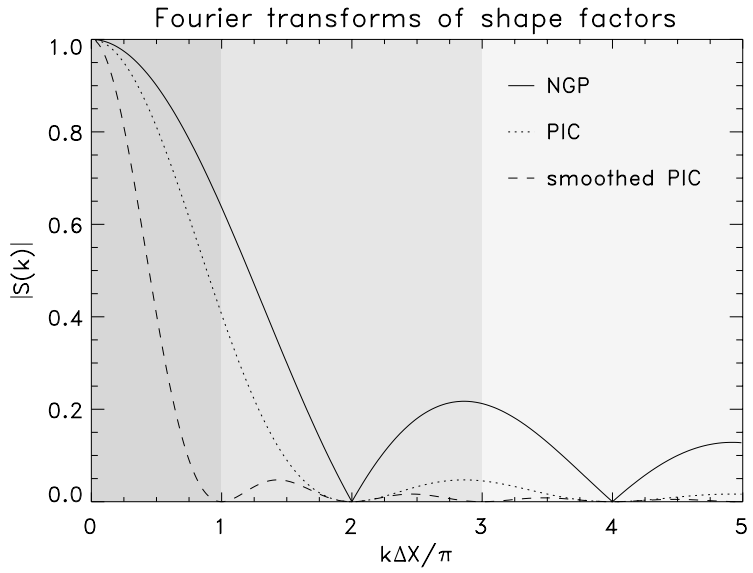

Figure 2.22: The Fourier-transformed shape factors of Fig. 2.21. The shaded areas indicate the first, second and third Brillouin zone, respectively. Note that the shape factor for smoothed PIC exists practically only in the first Brillouin zone.

the aliases are fed back into the system. In other words, the aliases become *coupled* through the grid. This was studied in detail by Birdsall and Langdon [1985] and was found to have an impact on the plasma dispersion.

The significance of alias coupling depends on how much of the alias power is folded back into the first Brillouin zone. According to Eqn. (2.111), this is determined by the shape factor $S(k)$. As a consequence of the smoothing that is applied in our code, the shape factor of our particles quickly decays beyond $k = \pi/\Delta X$ (Fig. 2.22). In our simulation, alias coupling is therefore not of great significance, and the plasma dispersion is described very well by Eqn. (2.117), which only considers the effect of the finite particle size.

2.8.2 Conservation of energy

A crucial test of the simulation code is to check in how far it respects the conservation of energy. There are three different forms of energy in the simulation:

1. The kinetic energy of the particles: $E_{kin} = \frac{1}{2}\sum_p m_p v_p^2$
2. The electric field energy: $E_{el} = \frac{\varepsilon_0}{2}\int E^2\, dV$
3. The magnetic field energy: $E_{mag} = \frac{1}{2\mu_0}\int B^2\, dV$.

Their sum $E_{tot} = E_{kin} + E_{el} + E_{mag}$ should not vary in time. If this is actually the case in our simulation, is first checked for electrostatic perturbations in a cold plasma. We simulate a single grid cell with periodic boundaries in all three directions. It is filled with one electron and one immobile ion that initially reside at the same location, so that there is no net charge density and no electric field at $t = 0$. Through the periodicity of the simulation domain this amounts to an infinitely large, homogeneous neutral plasma.

The electron "component" of this plasma is initialized with a velocity of $v_0/c = 0.1$. This give rise to electric fields in the direction of v_0, which tend to recover neutrality. Hence, what is to be expected from this configuration are 1D Langmuir oscillations at the plasma frequency.

Fig. 2.23 shows the temporal development of the three different energy forms and of their sum for a time span of $1000\Delta t$. They are normalized to the kinetic energy at $t = 0$, i. e. to $E_0 = m_e v_0^2/2$. As expected from Langmuir oscillations, electric and kinetic energy are anti-correlated and oscillate with the plasma frequency ($T_{pe} = 160\Delta t$ here). Since Langmuir waves are an electrostatic phenomenon, the magnetic field energy remains zero.

The total energy, however, is not perfectly conserved: There is no net increase in E_{tot} during the time interval shown, but it oscillates with ω_{pe} between $0.9997E_0$ and $1.002E_0$. In other words, the energy is conserved to within 0.2%. Considering that the "correct" energy value is periodically recovered, i. e. that there is no drift in E_0, this inaccuracy is tolerable.

Having seen that for electrostatic perturbations in a cold plasma the code respects conservation of energy quite well, we now focus on fully electromagnetic warm plasmas. A simulation grid of 20×20×20 cells with periodic boundaries is filled uniformly with an electron-ion plasma, so that $\rho = 0$ and $\mathbf{E} = 0$. The ions are made quasi-immobile by setting $m_i = 10^8 m_e$, and the electrons are initialized with a thermal velocity of $v_{e0}^{th}/c = 0.1$. The spatial and temporal resolutions are set to $r_x = 1$ and $r_t = 2$, respectively, which results in a plasma period of $T_{pe} = 300\Delta t$.

The evolution of the different energies is shown in Fig. 2.24. As above, the energies are normalized to the kinetic energy at $t = 0$, which makes up the total energy of the system E_0 at $t = 0$. Electrons and ions initially reside pairwise at the same location, and the "ionization" of the plasma takes place from $t = 0$ to $t = 1$: The electrons gain potential energy at the expense of their kinetic energy. This is the reason for the sharp change in E_{el} and E_{kin} during the first $100\Delta t$. The fraction of kinetic energy that is thereby transformed into field energy is only about $10^{-3}E_0$.

After the initial ionization drop, the kinetic energy remains rather constant and shows only minor fluctuations. Obviously, these are of electrostatic origin, because they are mirrored in E_{el} (in anti-phase) and not in E_{mag}. They practically cancel out in the total energy E_{tot}. Apart from that, both the electric and the magnetic field energies exhibit a linear increase of about $3 \cdot 10^{-4}E_0$ over the time span

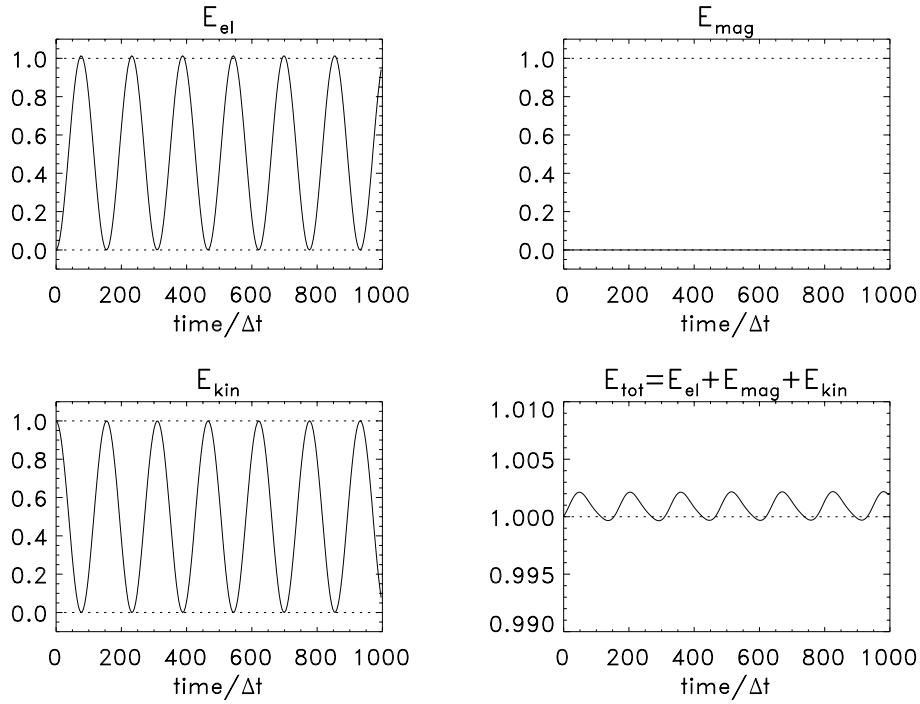

Figure 2.23: The cold plasma test case: temporal evolution of the different energy forms. (Note the different scale of the fourth panel.)

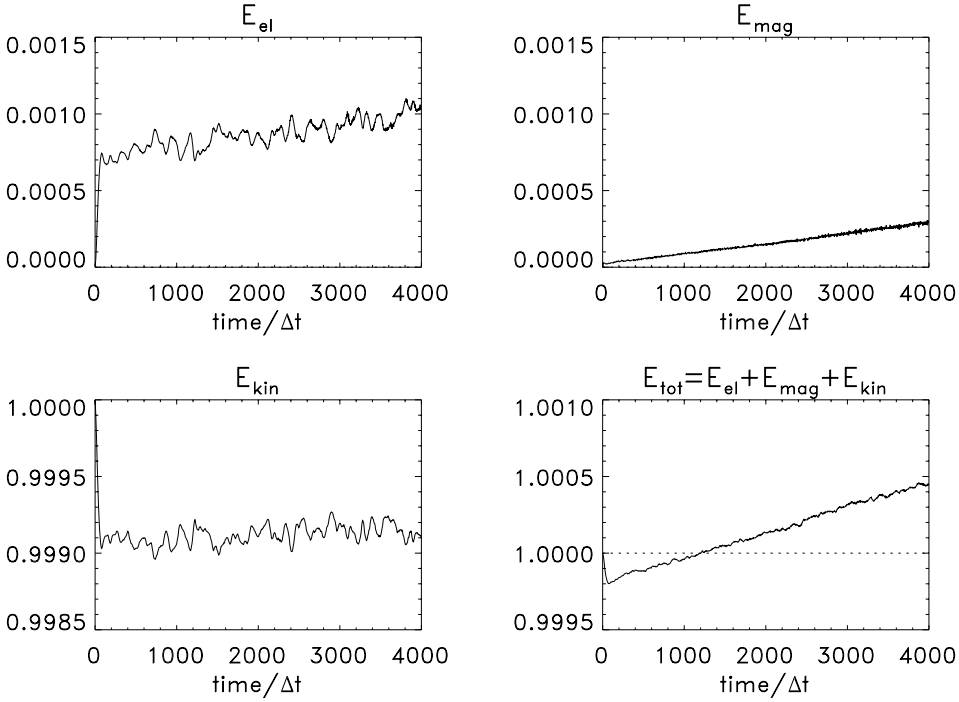

Figure 2.24: Temporal evolution of the energy forms for the warm plasma test case.

of $4000\Delta t$. This is *numerical* noise. In E_{tot} it adds up to a drift of $7 \cdot 10^{-4} E_0$ over the time interval shown, which corresponds to 0.2% after $10^4 \Delta t$.

The electromagnetic numerical noise did not show up in the first test of energy conservation, because it was covered by the much greater inaccuracies in the electrostatic field energy. On the other hand, these inaccuracies are also present in the warm plasma test. Here, however, they are not significant because the electric field contributes just a fraction of $10^{-3} E_0$ to the total energy of the system.

As a result, the two simulation runs carried out in this section revealed the following energy conservation characteristics of our code:

1. Short-term violations in the order of 0.2% of the total energy.

2. Long-term linear increase of 0.2% after $10^4 \Delta t$.

As both simulation runs can be regarded as prototypes for the simulation of cold and warm plasmas, respectively, these values are representative for any simulation scenario. They are acceptable for the majority of possible applications of our code, including ion thruster beam neutralization.

2.9 The code as a whole: ISOLDE

The preceding sections document the development of our simulation code. It is a 3D electromagnetic PIC simulation consisting of rather generic building blocks and thus suited for a variety of possible applications. As it is, however, "ISOLDE" – the **I**on engine **SOL**ver – is designed specifically for the simulation of both ion thruster beam neutralization and the solar wind interaction with the neutralized thruster beam.

To some extent, ISOLDE resembles Buneman's famous TRISTAN code: The field solve routine leap-frogs Maxwell's two curl-equations in time on the spatially staggered Yee lattice (Sec. 2.1). We make use of the charge-conserving current calculation method originally suggested by Villasenor and Buneman [1992], thus saving the integration of Poisson's equation (Sec. 2.2.2), and employ a standard relativistic particle push (Sec. 2.4).

Apart from some coding details like the current smoothing, which, in view of the later parallelization, we perform in a grid-based rather than a particle-based manner (Sec. 2.2.3), the main differences of ISOLDE as compared to TRISTAN concern the boundary conditions – both for fields and for particles.

Lindman's open boundaries [1973] for the electromagnetic field, which are employed in Buneman's TRISTAN code, were found to diverge in the presence of strong electrostatic fields in the boundary region. As such fields are an inherent

feature of the simulation of ion thrusters, we implemented a more robust boundary condition: a region of finite electrical conductivity to absorb outgoing electromagnetic waves. With a proper choice for the conductivity, these boundaries can reduce the reflection of outgoing waves to a tolerable level (Sec. 2.5.1).

The spatial separation between electron and ion source, which is a central characteristic of ion thrusters, poses a serious problem to the injection of particles into TRISTAN-like local electromagnetic field solvers. We have shown that a straightforward, uncompensated injection of a non-neutral plasma gives rise to artificial divergences in the electric field (Sec. 2.5.2). In order to avoid these divergences with ISOLDE, we have developed an injection technique that creates the charges *before* they are injected: the "generator". It allows to self-consistently inject non-neutral plasmas into local electromagnetic field solvers as it is needed for an appropriate simulation of ion thrusters.

Another feature that distinguishes ISOLDE from Buneman's TRISTAN code is the parallelization: Based on the so-called "taskfarm" concept, we have developed an efficient parallel version of ISOLDE, which will be described in the following chapter.

Chapter 3

The parallelization

Three-dimensional particle simulations as ours have high requirements towards computational power and usually have to be implemented on a supercomputer. The most powerful general purpose supercomputers today are of the massively parallel, distributed memory type. These machines consist of many processors, each with its own local memory, working either synchronously (SIMD – Single Instruction Multiple Data) or asynchronously (MIMD – Multiple Instruction Multiple Data) on a single problem.

The supercomputers at hand are two CRAY T3Es: one of the Institute of Scientific Computing in Braunschweig with 28 processors (PEs – Processing Elements) and the one of the Edinburgh Parallel Computing Centre with 344 PEs, 128 of which we were granted access to. Both are of the MIMD type and can be regarded as an ensemble of nearly identical, independent but interconnected computers.

Programming these computers in an efficient way generally requires special techniques or algorithms which differ from their corresponding sequential implementations due to the existence of distributed memory and the latency of accessing memory that is not local to a given processor. If the simulation algorithms of the sequential code are chosen to rely on local data only, these techniques reduce to adequately mapping the major data arrays onto the collection of processors and to providing the necessary inter-processor communication. In other cases, where data locality is not preserved, the efficient parallelization of a code might ask for a substantial reorganization of the sequential algorithm.

This chapter outlines how a parallel version of our simulation code was developed. We first implement a standard parallelization scheme, a so-called “domain decomposition”, which runs fairly efficiently on up to about 8 processors. For higher numbers of processors, however, its speed-up will turn out to be quite poor. After identifying the factors that inhibit a better speed-up, we develop a more sophisticated parallelization strategy, which better suits the characteristics of our code and runs with high efficiency on a massively parallel computer.

3.1 A straightforward parallelization

Running a parallel code on a MIMD computer gives access to the added up memories of the single processors – thus allowing larger simulations – and to their combined computational power in order to reduce simulation run times. In conjunction with the efficient use of these resources, a good parallel code meets two essential design criteria:

1. Scalability: The code should not be written for a fixed number of PEs. Instead, for a given problem, the simulation run time ideally decreases in proportion to the number of PEs used (linear speed-up). For varying problem sizes, it should remain constant when the number of PEs is chosen proportional to the problem size. Scalability is the central issue to be dealt with when writing a parallel program. The requirement of a well-scaling code obviously determines the parallelization strategy. An overall limit on the achievable scalability is imposed by the fraction of the code that cannot be run in parallel and by the time consumed by necessary inter-processor communication [Amdahl, 1967; Gustafson, 1988].

2. Portability: The code should not be written for a specific supercomputer. While one generally has to decide for a certain system architecture (e. g. SIMD or MIMD), portability between systems of different vendors but with similar architecture can be maintained by relying on standardized libraries for the inter-processor communication such as MPI (Message Passing Interface, [MPI, 1994; MPI-2, 1997]) rather than employing machine specific routines (like `shmem` on the CRAY T3E [Anderson et al., 1996]).

For the sake of a good scalability, the total workload of the simulation has to be distributed equally among the various processors, independent of the actual number of PEs used. Rather than a *functional* decomposition of the work, where every PE is assigned a specific task, this requirement suggests to let every PE carry out the *same* tasks, but on different parts of the data. The mapping of data onto the ensemble of processors is known as “domain decomposition”. It is the standard technique for the parallelization of plasma PIC simulations [Lyster et al., 1995; Wang et al., 1995].

For such simulations, domain decomposition amounts to subdividing the physical simulation volume into several domains, each of which is assigned to one PE [Lyster et al., 1995; Wang et al., 1995]. Each PE stores the data belonging to its domain, i. e. both the field quantities and the particles residing in this domain, and is responsible for their update. After each time step, the PEs have to communicate with each other in order to exchange field data and to trade particles that may have moved from their own domain to that of another PE.

Hence, the straightforward parallel code to be discussed in the following consists essentially of running the complete computational cycle of the sequential code

on every single PE and providing the necessary inter-processor communication. The code is a *single* program that is started on every PE. Nevertheless, individual commands on different PEs can be executed. This is done by referring to the PEs' unique id numbers that are determined upon program start.

For portability reasons, the inter-processor communication is programmed in MPI. Where such communication is required and how it is organized will be shown in the following. We thereby adapt the work of Wang et al. [1995], who successfully parallelized a plasma PIC code by virtue of the domain decomposition technique, to the needs of our simulation. Important extracts of the code are given in the form of pseudo-codes that resemble but do not exactly match FORTRAN syntax.

3.1.1 Domain decomposition

The total workload has to be distributed equally among the various PEs, in order for the code to run efficiently in parallel. A straightforward attempt to accomplish this is to subdivide the computational volume into domains of *equal size*. As a starting point, we therefore programmed a static, equidistant partition of the simulation volume that allows for decompositions in one, two or three physical dimensions, leading to slab-like, rod-like or block-like domains, respectively.

The total number of PEs to be employed for the simulation run and their spatial arrangement is determined up front by setting the number of PEs for each physical dimension `noprx`, `nopry`, `noprz` in the input parameter file (Fig. 2.19). With this information and its own id – `myid` – every PE can then determine its location within the overall PE arrangement, its PE neighbours and the bounds of its domain.

In order to illustrate how the individually correct settings are determined on each PE by referring to its id, we now show this localization procedure in some detail. For simplicity, a 1D decomposition in the x direction, i. e. with `nopry=noprz=1`, is considered. The processor ids run from `myid=0` to `myid=nopr-1`, where

```
nopr = noprx*nopry*noprz
```

is the total number of PEs used. The domain bounds in x are then computed by every single PE via

```
ilow  = nogx*myid/(1.*nopr)
ihigh = nogx*(myid+1)/(1.*nopr)-1
```

where `nogx` is the global number of grid cells in the x direction. Since with these definitions `ilow` of PE number `myid+1` is equal to `ihigh+1` of PE `myid`, this ensures a continuous 1D mapping of the complete grid on the series of PEs used.

The corresponding particle coordinate bounds for this segment of the grid are

```
xlow  = 1.*ilow
xhigh = 1.*(ihigh+1)
```

In y and z, the domain bounds are equal to the bounds of the global simulation volume

```
jlow  = 0
jhigh = nogy-1
klow  = 0
khigh = nogz-1
```

and, analogously to the x direction, the bounding particle coordinates are

```
ylow  = 1.*jlow
yhigh = 1.*(jhigh+1)
zlow  = 1.*klow
zhigh = 1.*(khigh+1)
```

Whether or not a PE is located next to a *global* boundary of the computational volume is stored in a Boolean array `bndry(1:6)` for all six possible boundaries. For the 1D decomposition regarded here, where all PE domains have global boundaries in $\pm y$ and $\pm z$, and only the first and the last PE have a global boundary in $-x$ and $+x$, respectively, this array would have to be set as follows:

```
bndry(1:6) = .TRUE.
IF myid .NE. 0 THEN bndry(1) = .FALSE.
IF myid .NE. nopr-1 THEN bndry(2) = .FALSE.
```

The neighbours of each PE, `nbour(1:6)`, which can only be on the $+x$ and/or $-x$ side, are then determined via

```
IF .NOT. bndry(1) THEN nbour(1) = myid-1
IF .NOT. bndry(2) THEN nbour(2) = myid+1
```

In this context, we note that *periodic* global boundary conditions in x could simply be realized by setting

```
bndry(3:6) = .TRUE.
bndry(1:2) = .FALSE.
IF myid .GT. 0 THEN nbour(1) = myid-1
               ELSE nbour(1) = nopr
IF myid .LT. nopr-1 THEN nbour(2) = myid+1
                    ELSE nbour(2) = 0
```

The complete localization procedure for a fully 3D domain decomposition, as it is used in our simulation, is somewhat more complicated and will not be given here.

Having thus determined the boundaries of its domain, each PE can now run the field solving routines in the same manner as in the sequential code, the only difference being the bounds of the update loop: Instead of `0`, `nogx-1`, `nogy-1`, and `nogz-1` as in the single-PE code, it runs on every PE from `ilow` to `ihigh`, from `jlow` to `jhigh`, and from `klow` to `khigh`.

While for the *interior* of the domain the sequential field solving routines carry over practically unchanged, the domain *boundaries*, however, require some caution. Depending on the position of the PE within the global simulation volume, two different cases have to be distinguished:

(i) The domain boundary is also a global boundary (`bndry(.)=.TRUE.`). In this case, the PE simply applies the global boundary condition at its domain boundary, i. e., in our simulation, it sets $\mathbf{E}_{tang} = 0$ in the outermost cell and modifies $\mathbf{E}$ in the adjacent conductive layer according to Eqn. (2.88).

(ii) The global grid continues beyond the domain boundary (`bndry(.)=.FALSE.`). The outermost cell of the PE domain would then have to be updated like an interior cell. As the field integration scheme of the sequential code uses central differences, this update relies on field information of the neighbouring cells (cf. Fig. 2.4). One of these cells – or, in the case of multi-dimensional domain decompositions, possibly more than one – lies, however, in the domain of another PE and is therefore not directly accessible.

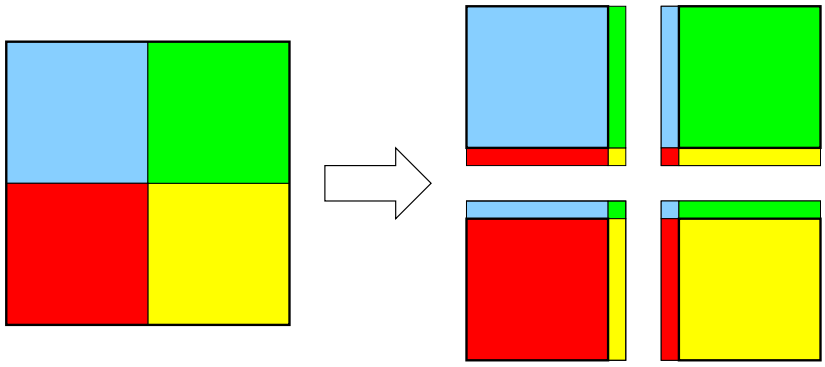

Figure 3.1: Domain decomposition in two dimensions: The ghost cells at the domain peripheries produce a double overlap between neighbouring domains.

For this reason, the domain of each PE is extended in all three dimensions by so-called *ghost cells* on either side, thus producing a double overlap between neighbouring domains (Fig. 3.1). The ghost cells are not processed, but just store the most recent field data of neighbouring PEs. In this way, each PE holds all the information needed to run the field solve routine over its whole domain, including

the domain boundaries. In order to contain the up-to-date field data, the ghost cells have to be filled by the neighbouring PEs after each field update. This is one of the stages where inter-processor communication is needed.

3.1.2 Inter-processor communication

For the sake of portability, the complete data exchange between different processors is programmed in MPI [MPI, 1994]. At four stages during the main loop of the code, such data exchange is required:

1. Filling the ghost cells after the magnetic field update.
2. Filling the ghost cells after the electric field update.
3. Accumulating the ghost cell current after the current calculation.
4. Exchanging the particles that leave their PE domain.

The ghost cell exchanges for the magnetic and the electric field work in the same way. Their pseudo-code for a fully three-dimensional domain decomposition reads as follows:

```
! field update in the interior of the domain:
CALL advance_E(ilow:ihigh,jlow:jhigh,klow:khigh)

! send outermost cell layer to neighbour in -x direction:
IF .NOT. bndry(1) THEN
   SEND E(ilow,jlow-1:jhigh+1,klow-1:khigh+1) TO
nbour(1)

! send outermost cell layer to neighbour in +x direction:
IF .NOT. bndry(2) THEN
   SEND E(ihigh,jlow-1:jhigh+1,klow-1:khigh+1) TO
nbour(2)

! receive fields from +x and store them in ghost cells:
IF .NOT. bndry(2) THEN RECEIVE FROM nbour(2) and
   store it in E(ihigh+1,jlow-1:jhigh+1,klow-1:khigh+1)

! receive fields from -x and store them in ghost cells:
IF .NOT. bndry(1) THEN RECEIVE FROM nbour(1) and
   store it in E(ilow-1,jlow-1:jhigh+1,klow-1:khigh+1)
```

```
! send outermost cell layer to neighbour in -y direction:
...
! send outermost cell layer to neighbour in +y direction:
...
! receive fields from +y and store them in ghost cells:
...
! receive fields from -y and store them in ghost cells:
...
! similar for ±z directions:
...
```

The send and receive operations are carried out by calling the corresponding MPI routines

`MPI_BSEND(data,n_elements,type,target_PE)` and
`MPI_RECV(rcv_buffer,n_elements,type,sender_PE)`,

which send the first `n_elements` of the array `data` of `type` to the PE `target_PE`, or receive `n_elements` from PE `sender_PE` and store them in `rcv_buffer`, respectively. For the first send and the first receive operations of the ghost cell update outlined above, these variables would be set as follows:

```
data = E(ilow,jlow-1:jhigh+1,klow-1:khigh+1)
target_PE = nbour(1)
n_elements = (jhigh-jlow+3)*(khigh-klow+3)
type = MPI_DOUBLE_PRECISION
```

and

```
rcv_buffer = E(ihigh+1,jlow-1:jhigh+1,klow-1:khigh+1)
sender_PE = nbour(2)
n_elements = (jhigh-jlow+3)*(khigh-klow+3)
type = MPI_DOUBLE_PRECISION
```

MPI actually offers several different send routines, the most common of which are `MPI_BSEND` (buffered send) and `MPI_SEND`. The former copies the data to be sent first into a local memory buffer before actually sending it, while the latter instantly executes the sending. The relevant difference between both routines is that `MPI_SEND` *does not return* before the *target PE* has received the data, whereas `MPI_BSEND` is terminated as soon as the data is copied into the *local* buffer [MPI, 1994]. In terms of the MPI nomenclature, `MPI_BSEND` is a "local" and `MPI_SEND` a "non-local" routine.

For the ghost cell update and the trading of particles, we exclusively use the local `MPI_BSEND`. This securely avoids *deadlocks*, which are common pitfalls in

programming parallel computers: Since every PE has at least one neighbour, all of them would call one of the send routines of the ghost cell update procedure above. If the non-local `MPI_SEND` was employed for the data transfer, the PEs would be stuck in this routine, because none of them would ever reach a receive operation that could release its neighbour out of the send routine.

There are three more details to the ghost cell update procedure that require some explanation:

1. The ghost cell update for one dimension (x, y or z) is completed before passing on to the next dimension, instead of, e. g., sending the fields first to *all six* neighbours before starting to receive.

2. Within the update for one dimension, however, the fields are first sent in *both* directions and then received.

3. The cell layer that is sent to the neighbouring PE extends from `low-1` to `high+1` in the respective dimensions, although the field update only runs from `low` to `high` (in all three dimensions). Hence, every PE sends some ghost cells that were not updated by itself.

Items 1 and 3 allow to significantly reduce the amount of inter-processor communication: In a fully three-dimensional decomposition, every PE has up to 26 neighbours, 6 of which are direct neighbours, 12 neighbours across the domain edges and 8 across the corners. With each of these neighbours, it would have to exchange ghost cell data, thus asking for 26 send and 26 receive calls after every field update.

If, however, the cell layers that are transferred to the respective neighbours are extended by one cell in each direction (item 1) and the transfer is done dimension-wise to the 6 direct neighbours only (item 3), the remaining 26 - 6 = 20 PEs receive the correct cell data automatically (Fig. 3.2). Hence, in this way, only 6 send and 6 receive calls between direct neighbours suffice to correctly deliver all the necessary data.

`MPI_RECV` is a "blocking" routine: It does not return before the data from the sender PE is actually received. Therefore, like all the other inter-processor communication to be discussed in the following, the ghost cell update procedure involves some degree of synchronization between the PEs: They have to wait until their neighbours are ready to send the required data. In order to reduce the useless waiting time of neighbouring PEs that possibly have less work to do, a PE should send the data required by others as early as possible. That is why within the ghost cell update of one dimension, the data is first sent in both directions before passing over to the receive operations (item 2).

From the necessary operations for the ghost cell update outlined above, it becomes apparent why it was desirable to implement a *local* field solver (cf. Sec. 2.1)

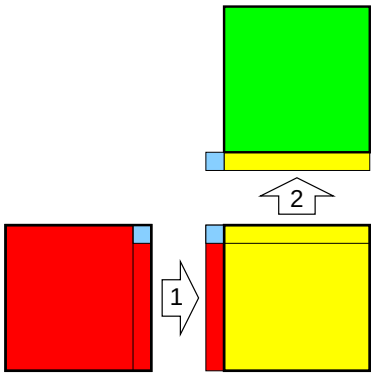

Figure 3.2: Our implementation of the ghost cell update allows the blue cell in the corner of the red PE to reach its destination without being transferred directly.

for the electromagnetic field update: Data from other PEs is required only on the periphery of the domain, while for the field update in the domain interior, all the necessary data is present on the respective PE.

The knowledge of the electromagnetic field in the cells adjacent to the domain boundary is not only necessary for the field update, but also for the gather routine: because the field values in these cells contribute to the force on particles residing close to the domain boundary. The implementation of the ghost cell update for the electromagnetic field as shown above therefore allows to take over the sequential gather and particle push routines without any changes.

This is also true for the parallel current calculation. It is identical to the sequential algorithm and simply runs over all the particles of a PE. Again, care has only to be taken with the domain boundaries: After the particle push, some of a PE's particles might have crossed its domain boundaries. These particles give rise to currents outside the domain. As the maximum displacement of a particle during one time step is restricted to less than one grid cell (cf. Sec. 2.7), it cannot produce currents on more than two layers of grid cells outside the domain. The convolution of the current in the smoothing procedure (Sec. 2.2.3) increases the maximum possible number of grid cell layers with non-zero current to three. The current in these grid cells has to be added to the corresponding grid cells of the neighbouring PEs.

Hence, for the current, a total of three ghost cell layers all around the domain is required. They are exchanged after the current calculation is complete, thereby following a similar procedure as for the exchange of electromagnetic ghost cells. While the latter, however, consisted of *copying* the field values of a PE's *regular* domain cells into the *ghost* cells of its neighbours, the current ghost cell exchange *adds* the currents of a PE's *ghost* cells to the *regular* cells of the neighbouring PEs.

An essential feature of the parallel code is that every PE holds all of the particles that reside within its domain boundaries `xlow`, `xhigh`, `ylow`, `yhigh`, `zlow`, and

zhigh. After the particle positions are updated in the push routine, however, some of them might have left the domain through one of its boundaries. If this boundary coincides with a global simulation boundary, then the particle is just removed from the list of existing particles of this PE; if not, the particle has to be passed on to the neighbouring PE. This is done in the particle exchange routine, which looks as follows:

```
! fill out-going buffers 1-6 for directions ±x, ±y, ±z and
! remove particles that leave global simulation volume:
DO LOOP over all particles
  IF x(particle) .LT. xlow THEN
    IF bndry(1) THEN remove particle from list
                ELSE put particle into buffer1
  ELSE IF x(particle) .GE. xhigh THEN
    IF bndry(2) THEN remove particle from list
                ELSE put particle into buffer2
  ELSE IF y(particle) .LT. ylow THEN
    IF bndry(3) THEN remove particle from list
                ELSE put particle into buffer3
  ELSE IF y(particle) .GE. yhigh THEN
    IF bndry(4) THEN remove particle from list
                ELSE put particle into buffer4
  ... similar for zlow and zhigh ...
  ENDIF
END LOOP

! exchange particles in ±x direction:
IF .NOT. bndry(1) THEN send buffer1 to nbour(1)
IF .NOT. bndry(2) THEN send buffer2 to nbour(2)
IF .NOT. bndry(2) THEN receive incoming particles from nbour(2)
IF .NOT. bndry(1) THEN receive incoming particles from nbour(1)

! check incoming particles and add them to out-going buffers if necessary:
DO LOOP over incoming particles
  IF y(particle) .LT. ylow THEN
    remove particle from incoming list
    put particle into buffer3
  ELSE IF y(particle) .GE. yhigh THEN
    remove particle from incoming list
    put particle into buffer4
  ... similar for zlow and zhigh ...
  ENDIF
ENDDO

! exchange particles in ±y direction:
```

```
IF .NOT. bndry(3) THEN send buffer3 to nbour(3)
IF .NOT. bndry(4) THEN send buffer4 to nbour(4)
IF .NOT. bndry(4) THEN receive incoming particles from nbour(4)
IF .NOT. bndry(3) THEN receive incoming particles from nbour(3)

! check incoming particles and add them to out-going buffers if necessary:
DO LOOP over incoming particles
  IF z(particle) .LT. zlow THEN
    remove particle from incoming list
    put particle into buffer5
  ELSE IF z(particle) .GE. zhigh THEN
    remove particle from incoming list
    put particle into buffer6
  ENDIF
ENDDO

! exchange particles in ±z direction:
IF .NOT. bndry(5) THEN send buffer5 to nbour(5)
IF .NOT. bndry(6) THEN send buffer6 to nbour(6)
IF .NOT. bndry(6) THEN receive incoming particles from nbour(6)
IF .NOT. bndry(5) THEN receive incoming particles from nbour(5)

append incoming particles to particle list
```

Similar to the ghost cell update, the trading of particles is carried out dimension-wise: The particle exchange of one dimension is completed before starting the exchange in the next dimension. Thereby, the *incoming* particles are checked on whether they really belong to the receiving PE, or have to be passed over to another neighbour. Again, this allows to deliver the particles correctly to the up to 26 PEs to which they can have possibly moved, with just 6 communications, thus saving a huge amount of inter-processor communication. Finally, those of the incoming particles that do belong to the respective PE are appended to its list of existing particles.

The particle trade as just described ensures that all of the *existing* particles are always located where they ought to. That this is the case also for the *newly injected* particles is made sure by centralizing the creation of new particles at one PE: One of the PEs that are employed for a simulation run is not assigned a physical domain, but is set aside for the injection of new particles. This “master PE” generates the random numbers for position and velocity of new particles and delivers each new particle according to its injection location to the respective processor to whom it belongs. Apart from that, the master operates the “generator” (see Sec. 2.5.2) for the consistency of particle injection and electromagnetic field. This involves calculating the generator current and delivering it to those PEs that have a share in the generator area.

3.1.3 Performance analysis

The main loop of the parallel code after integrating the inter-processor communication is shown in Fig. 3.3. In blue are the adapted routines of the sequential code. Their order is not changed as compared to that of the sequential version and greatly determines the locations of the communication routines (red): A ghost cell update has to be executed *after* the quantity in question has been advanced and *before* it is needed as the input for another routine.

The first $\mathbf{B}$-field ghost cell exchange, for instance, has to be executed after the update of $\mathbf{B}$ and before the $\mathbf{B}$-field is needed for the relocation procedure. This leaves only one possible location for this routine (cf. Fig. 3.3). For the positioning of the other communication routines, we had some more freedom, and, where possible, tried to arrange them in groups. As every communication involves some degree of synchronization between the PEs, this grouping reduces the number of inevitable synchronizations and thus the idle time of the PEs.

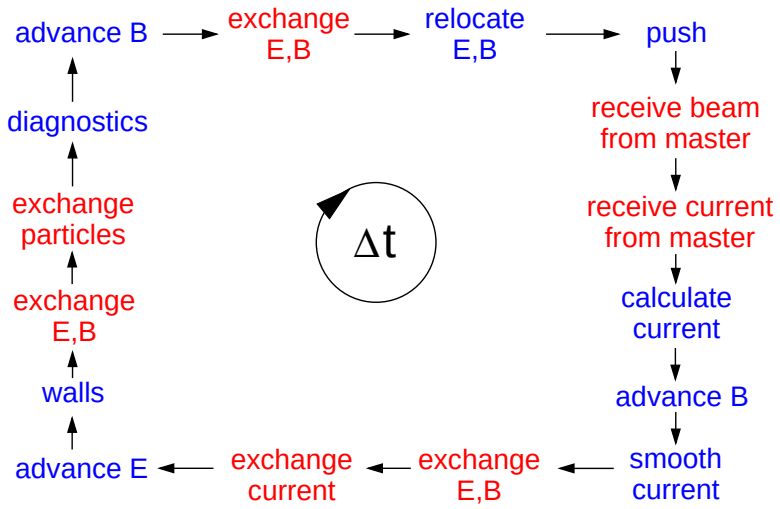

Figure 3.3: The main loop of the slave PEs. The *diagnostics* routine involves a synchronization of all PEs and is not called every time step. The routine *walls* applies the global boundary conditions for the electromagnetic field.

The main loop of the master PE is significantly simpler (Fig. 3.4). The master's communication with the working PEs ("slaves") consists of sending them the newly injected particles and the current contribution of the generator. On the side of the slaves, this communication with the master separates the two computationally very expensive routines – particle push and current calculation, which both involve loops over all particles. In our straightforward parallelization, this separation is necessary, because the new particles contribute already to the new current, but must not yet be advanced by the standard push routine (see Sec. 2.5.2). This will later be changed in the optimized version of the parallel code (Sec. 3.2).

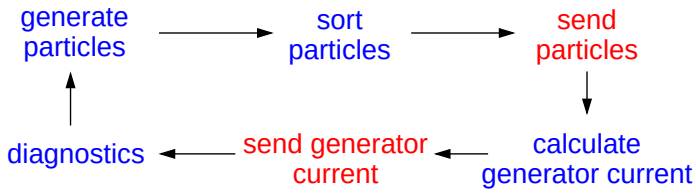

Figure 3.4: The main loop of the master PE. Similarly to the main loop of the slaves, the *diagnostics* routine is not called every time step.

At the end of both loops, about every 20 time steps, a diagnostics routine is called to save field and particle data. This routine also conducts a synchronization of *all* PEs (via `MPI_REDUCE`, [MPI, 1994]), i. e. each PE is forced to stop at that point until every other PE has reached it. This regular synchronization is necessary in order to refrain the master PE, which has much less work than the slaves, from "running away". Hence, the master will spend a considerable amount of time at this synchronization point waiting for the slaves to catch up.

With our MPI-based straightforward parallelization as described in the foregoing, the code runs on every parallel computer where MPI is installed. In order to assess how efficiently it works on these computers, there are generally two options [Wang et al., 1995; Schüle, 2000]:

1. Fixed problem size analysis: The same simulation is run with different numbers of PEs. If T_n is the total simulation time needed when running the simulation on n PEs, then the parallel efficiency is $P_n^{fix} = T_1/(nT_n)$.

2. Scaled problem size analysis: The problem size is increased in proportion to the number of PEs used, and the parallel efficiency is calculated as $P_n^{scal} = T_1/T_n$.

Which one of these definitions of parallel efficiency is adopted depends on the actual objective of the parallelization. In most cases, the scaled analysis will give the more appropriate assessment, because one usually does not parallelize a code just in order to solve the same problem more quickly, but to take advantage of the larger memory of multiple PEs for solving larger problems.

For the sake of simplicity, we carried out a fixed-problem size analysis. This will most certainly give a lower efficiency than a scaled analysis: With increasing numbers of PEs, the actual computational workload of each PE decreases, while the useless time spent on communication stays essentially the same and thus becomes increasingly important [Gustafson, 1988].

Fig. 3.5 shows the input parameter file for the speed-up measurements. The simulation box measured $100 \times 60 \times 60$ grid cells and the electron-ion beam injected on the left side had a diameter of 20 cells. The number of time steps was chosen such that by the end of the simulation the beam almost reached the right

```
------------Number of processors in x
4
------------Number of processors in y
2
------------Number of processors in z
2
------------Number of species (including electrons)
2
------------Super-particle density in the beam (N_S)
20
------------Number of grid-points in x-direction
100
------------Number of grid-points in y-direction
60
------------Number of grid-points in z-direction
60
------------Number of time steps
1800
------------Time step for diagnostics
20
------------r_x (dx=r_x*l_Debye)
1.D0
------------r_t (dt=dt_max/r_t)
1.5D0
------------Conductivity of absorbing boundaries (sigma')
0.05D0
------------Thickness of conductivity layer
5
------------Electron gyro period / electron plasma period
0.D0
------------Electron thermal velocity / c
0.1D0
------------Electron beam velocity / c
0.D0
------------Electron beam width (in grid cells)
20
------------Beam spacing (in grid cells)
-20
------------SPECIES2: charge/|e|
1.D0
------------SPECIES2: mass/m_e
250000.D0
------------SPECIES2: number density/electron density
1.D0
------------SPECIES2: temperature/electron temperature
1.D0
------------SPECIES2: beam velocity / c
0.1D0
------------SPECIES2: beam width (in grid cells)
20
------------End of inppar_speedup
```

Figure 3.5: The parameter file *inppar* for the speed-up measurements.

boundary. This gives a workload distribution between particles and grid that is representative for future production runs.

The results of the speed-up measurements are displayed in Fig. 3.6, which shows the parallel efficiency P^{fix} as a function of the number of PEs. For better comparison, this number does not include the master PE but only the slaves, i. e. those PEs that actually share the workload of the simulation domain among themselves. Moreover, the reference problem is too big to run on a single PE. Deviating from the original definition of the fixed parallel efficiency (see above), we therefore normalized it to the simulation time of 4 PEs:

$$P_n^{fix} = 4T_4/(nT_n)\,. \tag{3.1}$$

As described above, our code allows for a domain decomposition in up to three dimensions. The measurements were therefore carried out for a variety of processor arrangements and were grouped according to the number of PEs in the $y-z$ plane: 1 PE (i. e. a decomposition only in the x direction, crosses in Fig. 3.6), 2×2 PEs (triangles), and 3×3 PEs (squares).

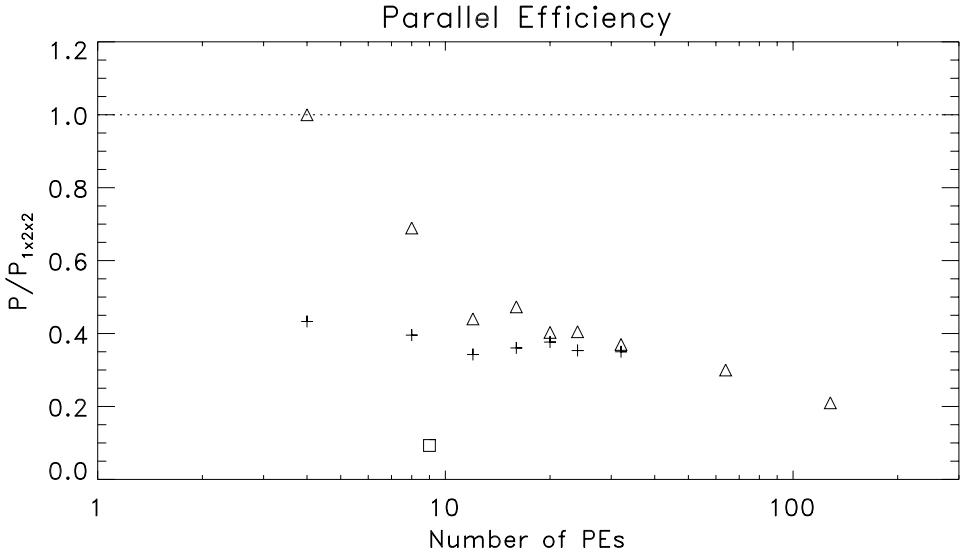

Figure 3.6: The parallel efficiency for the reference simulation of Fig. 3.5. Different symbols correspond to different PE arrangements in the y-z plane: 1 PE (+), 2×2 PEs (△), and 3×3 PEs (□). The efficiency is normalized to the simulation run with 1×2×2 PEs, and the number of PEs does not include the master.

Two main features of the parallel code can be deduced from the measurements displayed in Fig. 3.6: (i) for less than 20 PEs, the parallel efficiency depends strongly on the actual processor arrangement, and (ii) the efficiency for PE numbers beyond 8 is rather poor. It decreases monotonously to 0.2 for 128 PEs.

In order to pursue the first of these observations, we compare the profiles of two simulation runs with equal numbers of PEs but different PE arrangements in Fig.

3.7. The bars in this figure indicate the times spent in the most time-consuming routines of the code for simulation runs with 4×1×1 and 1×2×2 PEs, respectively. They are *the sum* over all PEs (including the master) and are grouped into two categories: computing routines and communication routines.

As can be seen from Fig. 3.7, the most expensive computing routines are the particle push and the current calculation, i. e. those that operate on the particles. Among the grid-based routines, the electric field update takes longest, because it accesses all three grid quantities (current, electric and magnetic field). The time spent on each of these computing routines turns out to be rather independent of the PE arrangement. Since the *total computational workload* of the simulation is the same for both runs, this does not come as a surprise. There is, however, a large discrepancy between both runs in the time consumption of the communication routines. The 4×1×1 run takes more than 10 times longer for the receiving calls of the guard cell exchanges and the particle trade, and about twice the time on the synchronization call of the diagnostics routine! It is these differences that are responsible for the 4×1×1 PE arrangement being much slower than the one with 1×2×2 PEs.

The time spent in the synchronization call `MPI_REDUCE` is the waiting time of those PEs that arrive first to this call. As stated above, the master PE has only little work to do as compared to the slaves and will therefore spend most of its time waiting for the others at this synchronization point. Hence, being one PE out of a total number of 5, the master accounts already for almost $1/5$ of the total simulation time that will be spent in `MPI_REDUCE`. Indeed, in both simulation runs, `MPI_REDUCE` takes about 20% of the total time. For increasing numbers of PEs, this fraction can be expected to reduce as $1/n$, with n being the total number of PEs used, and is therefore not a matter of concern for running the code on massively parallel computers.

Much more worrying is, however, the great deal of time spent in the receiving operations. They are called only by the slave PEs. As stated earlier, these operations are blocking: A PE does not return from them until it has received the requested data completely. Hence, those times are mostly *waiting times*, the mere reception of the data once it was sent making only a negligible contribution. The times spent in the receive calls are therefore a good indicator of the *workload imbalance* among the slaves and, in this case, clearly reveal that the work is much better distributed in the 1×2×2 run than it is for 4×1×1 PEs.

As far as the grid-based routines are concerned, the uniformly-sized domains of our equidistant decomposition make sure that their work is distributed equally among the various PEs, independent of the PE arrangement. These routines do therefore not give rise to any imbalance in the workload. The particle associated work, however, is well-balanced only for a few special PE arrangements. In general, the strong inhomogeneity of the particle distribution, with the beam emerging from the central area of the left simulation boundary and growing in the x direction, causes the share of the beam to differ considerably from domain

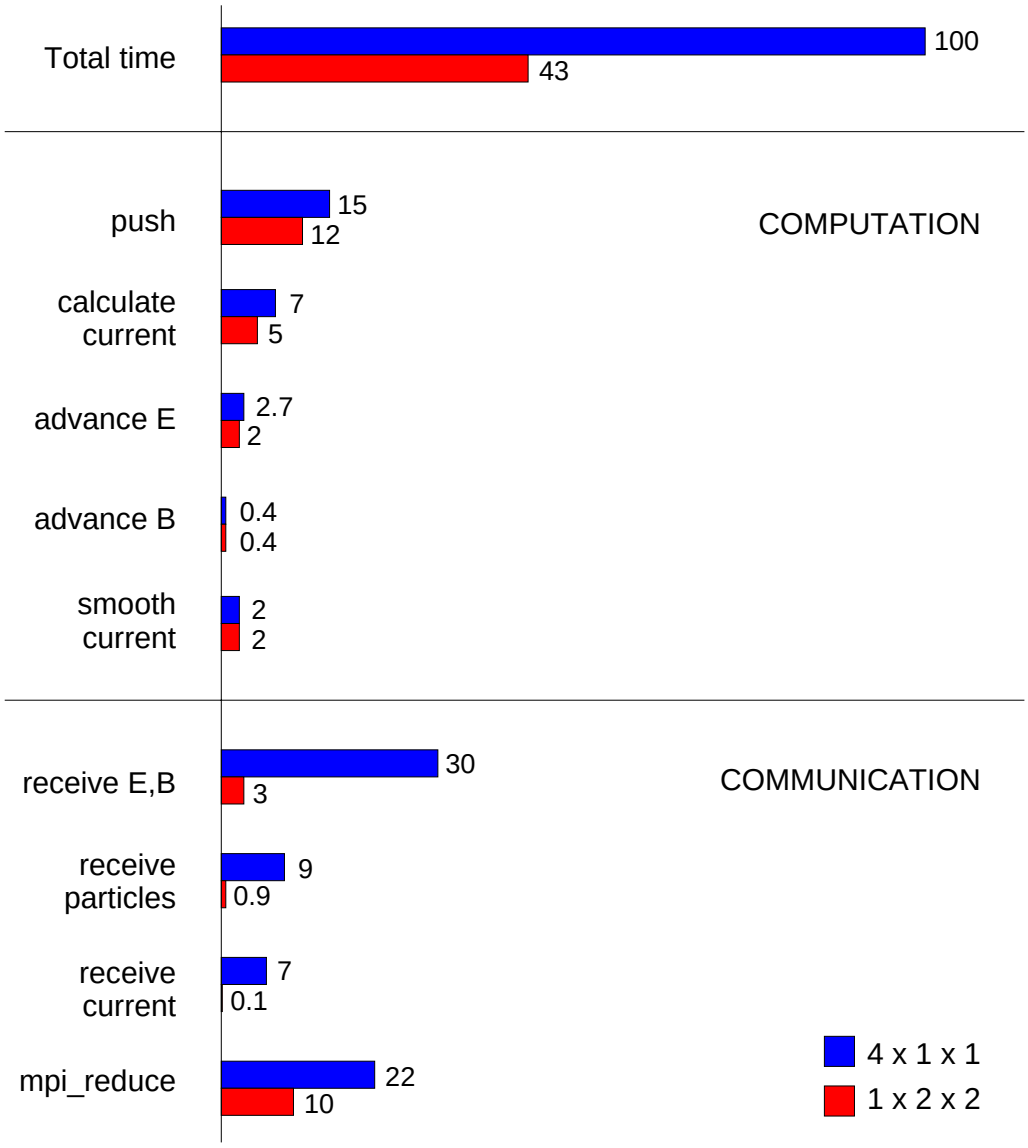

Figure 3.7: The profiles of the simulation runs with $4\times1\times1$ PEs (blue) and $1\times2\times2$ PEs (red). The times shown are the sum over all PEs, including the master, and are normalized to the total time of the $4\times1\times1$ run. Note the vast amount of time spent in the receiving operations when running on $4\times1\times1$ PEs. The time of *mpi_reduce* is mainly waiting time in the synchronization call of the diagnostics routine and is to the greatest part caused by the master PE. The ratio between particle associated work (*push* and *current calculation*) and grid-based work (*advance E*, *advance B*, *smooth current* and others) is around 4. (Not all routines are shown.)

to domain. Those PEs with domains close to or inside the beam area are responsible for significantly more particles than those with domains farther off. Hence, it is the very strong inhomogeneity of the particle distribution that generates the workload imbalance.

Due to the symmetry of the simulation geometry, the decomposition with $1\times2\times2$ PEs is one of the few arrangements with a fairly well-balanced particle workload: Every PE hosts roughly a quarter of the total number of particles during the *whole* simulation. This is not the case for the $4\times1\times1$ arrangement: Only at the very end, when the beam extends over the whole simulation volume in the x direction, there are approximately the same number of particles in each of the four domains. During the first quarter of the simulation time, however, practically all particles reside on the outermost PE on the left. Only later, also the second, third and fourth PE get a share of the beam. This imbalance of the particle distribution makes the $4\times1\times1$ decomposition much slower than the one of $1\times2\times2$ PEs.

An even more extreme case of particle imbalance is the simulation run with 3×3 PEs in the $y-z$ plane (Fig. 3.6). Here, the central PE comprises practically the complete beam during the whole simulation, while the neighbouring eight PEs are nearly void of particles and spend most of their time waiting on the central PE to finish its particle work.

Such unfavourable PE arrangements are, of course, to be avoided for production runs. However, for more than four PEs, there is no way of equidistantly decomposing the simulation volume such that each domain has approximately the same number of particles during the whole simulation. For total PE numbers of more than four, a workload imbalance is therefore inevitable. Moreover, this imbalance becomes severe for greater numbers of PEs: With the domains becoming smaller, the discrepancy in the domains' beam shares is bound to increase, as some domains might be completely submerged in the beam area, while others are far off and might not even get a single particle during the whole simulation run. This explains the rather poor parallel efficiency of the code for higher PE numbers.

As a result, the parallel code is fairly efficient when running on up to about 10 PEs. Due to the strong imbalance of the particle associated work, however, the code in its present form is not suitable for larger production runs. In order to simulate bigger systems requiring memory and computational power of 64 or more PEs, the parallelization strategy has to be optimized.

3.2 The optimized parallelization

According to the performance analysis of the last section, the major shortcoming of the straightforwardly parallelized code is the non-uniformity in the distribution of the beam particles among the PE domains. Since this non-uniformity is a consequence of the equidistant domain decomposition, the first idea one might come up with to mitigate the workload imbalance is to abandon the regularity

of the equidistant decomposition and to introduce a decomposition that allows for smaller domains in the center and larger ones further outside (Fig. 3.8). With such variably-sized domains, whose boundaries could, moreover, be made to adjust themselves during the simulation according to the momentary distribution of particles, one might manage to distribute all particles equally among the PEs and thus to remove the imbalance in the particle workload.

Such a non-equidistant decomposition would, however, introduce an imbalance in the work of the grid routines, because their computational cost is proportional to the number of grid cells, i. e. to the size of the domain. As a result, allowing for the domains to be of variable size would just shift the imbalance from the particle routines to the grid routines. Neither would it help to choose the domain sizes such that the *sum* of particle and grid work is the same for all domains: During one time step, there are several points where inter-processor communication is necessary (Fig. 3.3). In order to reduce the waiting time at these points, the blocks of work between any two of them have to be well-balanced. Hence, both particle work and grid work have to be balanced individually!

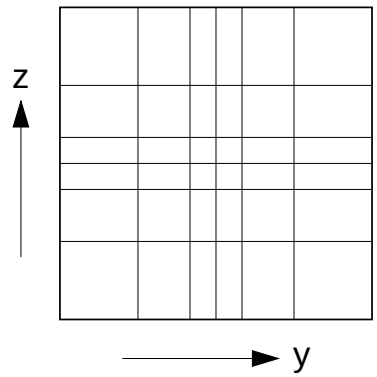

Figure 3.8: A non-uniform domain decomposition in the y-z plane with smaller domains in the center and larger domains farther outside. Such a decomposition with dynamically self-adjusting walls would allow to distribute the particles equally among all PEs.

For the grid routines, this means that one has to stick to the equidistant decomposition, because this is the only way of balancing their work. In order to balance also the particle work, Liewer and Decyk [1989] suggested to introduce a *secondary* decomposition that adjusts itself dynamically to accomodate roughly the same number of particles in each domain. The secondary decomposition is also a physical division of the simulation box into sub-volumes, but it is independent of the primary equidistant decomposition and is only determined by the current particle distribution.

Named “General Concurrent Particle-in Cell” algorithm (GCPIC), Liewer and Decyk’s dynamic load balancing with two independent decompositions clearly generates the same amount of particle and grid work on every PE. Its drawback

is, however, that it does not preserve data locality and thus involves a great deal of inter-processor communication: With only a single decomposition as in our straightforward parallelization, every PE has all the data at its disposal that is needed to update particles and fields. An exchange of particle or field data is necessary only at the domain boundaries (via ghost cells, cf. Sec. 3.1.2). This changes dramatically for the GCPIC. In order to update the particles in its particle domain, a PE has to gather all the field data of the corresponding region of the simulation box from the various PEs whose primary domains intersect with this region. Similarly, after calculating the current from the particle movement, it has to send appropriate portions of the total current array to these PEs.

Liewer and Decyk [1989], Ferraro et al. [1993], Lyster et al. [1995], and Wang et al. [1995] implemented the GCPIC into electrostatic and electromagnetic PIC simulations. They found that due to the communication overhead, the dynamic load balancing is counterproductive compared to a single static equidistant decomposition, unless the particle work greatly dominates over the grid work. In their 3D electromagnetic simulations, where the particle push was about 100 times more expensive than the field solver, Wang et al. [1995] achieved a parallel efficiency of nearly 100% by using the GCPIC. The benchmarking of Ferraro et al. [1993], however, who used a code with a ratio between particle work and grid work of about 3, showed the GCPIC to be actually inferior to a static equidistant decomposition.

For our reference simulations of the last section, the computational cost of the particle associated work dominated over the grid work by a factor of about 4 (Fig. 3.7). Hence, the GCPIC approach will probably not significantly improve the parallel efficiency of our code, and another way of removing the strong imbalance of the particle workload has to be envisaged.

A more promising option than the GCPIC is to implement a so-called “taskfarm”: The single static equidistant decomposition of our straightforward parallelization is kept, but additionally, on every PE, all the particles are now organized into a certain number of smaller groups each of which forms one “task”. Once it is time to push the particles, these groups are traded among *all* PEs. Whenever a PE has finished the work on its own particles, it grabs a particle group of another PE, processes it and writes it back. Hence, PEs with less particles take over parts of the work of other, busier PEs.

The transfer of particle groups from and to their host PE has to happen without the host PE’s involvement in order not to hold it off from its own work. Such a “one-sided communication”, as it is called in the MPI jargon, which allows one PE to freely access the memory of another, is available in the extension of the MPI standard [MPI-2, 1997].

In order to manage the taskfarm, the master PE of our straightforward parallelization is now assigned the additional duty of being the “taskmaster”: It keeps a list of all the tasks, coordinates the work progress and makes sure that no PE is released before all tasks are done.

A taskfarm of this kind – without a taskmaster, though – has been employed e. g. by Carretti [1998] in his simulation of gravitating galaxy clusters. In plasma PIC simulations, however, the taskfarm concept has not been introduced so far. The following three sections will therefore show in some detail how we implemented this concept into our plasma simulation.

3.2.1 Managing the taskfarm

The essential prerequisite of the taskfarm is that every PE organizes its particles into a certain number of groups. In our implementation, this number, `no_slabs`, is the same on all PEs. The particle groups are formed by sorting the particles according to their z coordinate into `no_slabs` slabs of equal thickness. Why the sorting is done in the z direction will be made clear at a later stage. Each of these slabs, together with the corresponding particle group, forms one task, which can be identified by two numbers: the id of the PE to which the group belongs and its slab number.

As far as the general structure of the code is concerned, the greater part of the main loops for master and slaves (Figs. 3.3 and 3.4) carry over with no change

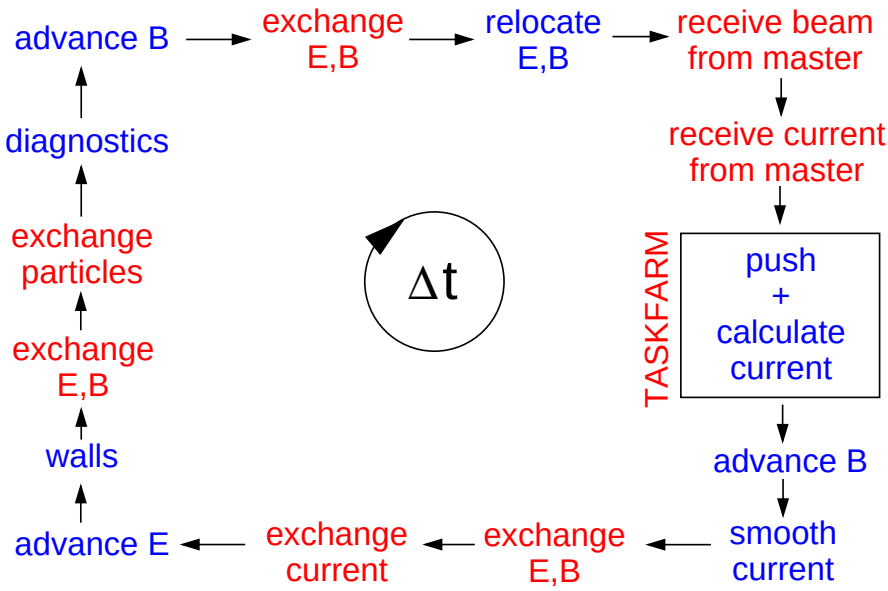

Figure 3.9: The main loop of the slaves in the optimized version of the parallel code. After receiving the generator current from the master, all PEs now synchronize and enter the taskfarm. Here, the main particle routines *push* and *current calculation* are carried out. Although not working on exactly the same particles (Secs. 2.5.2 and 3.1.3), these two routines are now executed one after the other, which requires some care (Sec. 3.2.2).

(see Figs. 3.9 and 3.10). Instead of starting the particle push routine, however, all PEs now synchronize (with `MPI_ALL_REDUCE`, [MPI, 1994]) and enter the central routine of the taskfarm.

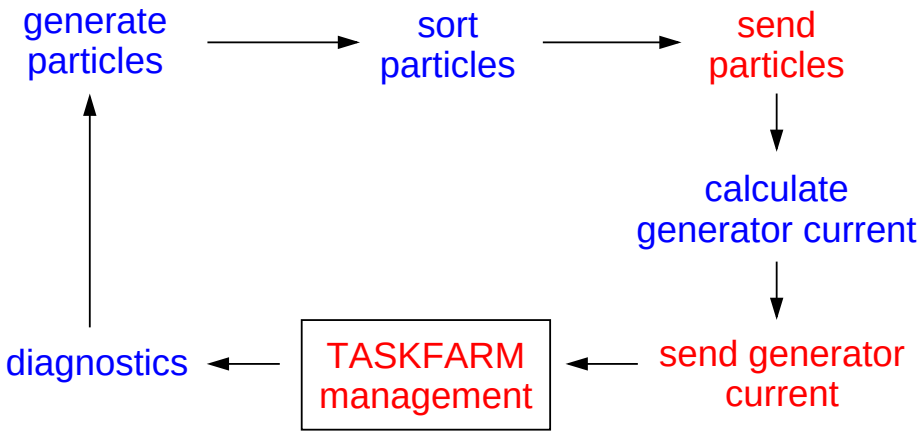

Figure 3.10: The main loop of the master in the optimized version of the parallel code. It now includes the management of the taskfarm.

Here, the slaves first report the size of the tasks in their own domain (i. e. the number of particles in each slab) to the master PE. After that, they enter a loop where they wait for messages from the master. These messages contain the task which they will then set out to do, or a signal to abandon the loop and to resume their own work. In pseudo-code, the slaves' part of the taskfarm management routine looks as follows:

```
IF myid.NE.master THEN
   report particle_numbers(1:no_slabs) to master
   receive work_package from master
   WHILE workpackage(1).NE.-1

      ! do the task given by the master:
      CALL do_task(work_package)
      ! tell master that work is finished and receive new task:
      send work_package(1:2) to master
      receive work_package from master
  ENDWHILE
ENDIF
```

The work package consists of 13 entries `work_package(1:13)` that contain all the necessary information to specify the task and to locate the required data in the remote PE's memory:

	content
work_package(1)	target PE
(2)	number of slab on target PE
(3:4)	lower and upper x bounds of slab
(5:6)	lower and upper y bounds of slab
(7)	klow of target PE
(8:9)	lower and upper z bounds of slab
(10:11)	actual numbers of electrons and ions
(12:13)	old numbers of electrons and ions

How this information is used to carry out the given task in the routine do_task is shown in the next section. A "-1" in the first entry of the work package is the signal for the slave to exit the loop (see above). Once all the tasks are done, this signal is sent by the master at the end of the taskfarm management routine:

```
IF myid.EQ.master THEN
  work_status(:,:)=0
  finished=.FALSE.
  found=.FALSE.
  WHILE .NOT. finished
    wait for message from any PE
    IF message_type=1 THEN
      ! slave PE sender reports work on its own domain
      save particle_numbers of sender in part_n(sender,1:no_slabs)
      work_status(sender,1:no_slabs)=1
      DO LOOP over slabs
         IF part_n(sender,slab)=0 THEN work_status(sender,slab)=3
      ENDDO
      ! give some of slave's own work back:
      work_package(1)=sender
      CALL look_for_work(work_package,found)
      fill work_package with necessary information
      send work_package to sender
      work_status(work_package(1:2))=2
    ELSE
      ! slave has finished a task
      receive work_package(1:2) from sender
      work_status(work_package(1),work_package(2))=3
      ! find new task, first on sender's own domain:
      work_package(1)=sender
      found=.FALSE.
      CALL look_for_work(work_package,found)
      IF found THEN
```

```
        fill work_package with necessary information
        send work_package to sender
        work_status(work_package(1:2))=2
      ELSE
       ! look for tasks on other PEs:
        DO LOOP over all slave PEs except sender
          CALL look_for_work(work_package,found)
        ENDDO
        IF found THEN
          fill work_package with necessary information
          send work_package to sender
          work_status(work_package(1:2))=2
        ELSE
         ! check if all the work is done:
          finished=.TRUE.
          DO LOOP over all slave PEs
            DO LOOP over all slabs
               IF work_status(id,slab).NE.3 THEN finished=.FALSE.
            ENDDO
          ENDDO
        ENDIF
      ENDIF
    ENDIF
  ENDWHILE
  ! send termination signal:
  work_package(1)=-1
  send work_package to all slave PEs
ENDIF
```

In the two-dimensional array `work_status(1:nopr,1:no_slabs)` the master keeps track of the work progress:

		meaning
`work_status`	`= 0`	work has not yet been reported
	`= 1`	work has been reported and is to be done
	`= 2`	work is in progress
	`= 3`	work is done

Many of the slabs do not contain any particles, especially in the beginning of the simulation. Therefore, the master PE immediately checks for empty slabs after receiving the work report from a slave and marks them as being done already. This helps to avoid unnecessary communication.

Moreover, in order to save data transfer between PEs, the master gives preference to the PE's *own* work when looking for a new task. Only once all the

work of its own domain is finished, a PE receives tasks on other PEs. The routine `look_for_work`, which searches for a new task, simply runs over the array `work_status` and looks for the *largest* piece of work, i. e. the slab with most particles. It returns `found=.TRUE.` if it has succeeded in finding a new task.

Tackling the tasks in decreasing order of size avoids that, towards the end, some PEs have to process a couple of big tasks while the other PEs are just waiting. Hence, this order is more likely to produce a well-balanced work distribution than any other or no order.

3.2.2 Execution of the task

Once a slave PE receives a task from the master, it sets out to perform it in the routine `do_task`. In our simulation, a task comprises all the work on a specific group of particles: both particle push and calculation of the current. If the task turns out to belong to the PE's own domain, then all the necessary data to perform it is already present, and the PE can right away carry out the particle push and the current calculation on the respective group of its own particle population.

As stated earlier, push and current calculation do not operate on exactly the same particles: The newly injected particles coming in from the master PE must not yet be pushed but contribute already to the current. This is why in the sequential code and in the straightforward parallelization, the particle push occurs before the injection of new particles, while the current calculation is executed thereafter. For the taskfarm, however, it is desirable to bundle the two particle-based routines into a single task. Therefore, the work package provides two numbers of particles: an old one for the particle push, and the actual number including already the newly injected particles. In this way, the particle push can be executed after the injection of new particles and thus directly before the current calculation (Fig. 3.9).

If a PE receives a task on another PE's domain, as a first step, it has to retrieve all the necessary data to perform this task from the target PE, namely (i) the particles of the slab specified in the task and (ii) the relocated electromagnetic field of this slab for the computation of the particle forces. Having copied this data into its own local memory, the PE can then execute the common particle push and current calculation routines. Finally, the updated particles are put back into the target PE's memory, and the calculated current contribution is added to the appropriate portion of the current array on the target PE.

All this is done without any involvement of the target PE, in order not to hold it off from its own work. To enable such a "one-sided communication" with free access to another PE's memory requires some preparation. For every portion of the memory that is intended to be accessible by other PEs, a so-called "window" has to be opened collectively by all PEs. Data can then simply be transferred to or from these windows with the `MPI_PUT` and `MPI_GET` routines of MPI-2 [MPI-2,

1997]. Among other parameters, a call to these routines has to specify the target PE, the window to be accessed, the size of the array to be transferred and an offset. The latter is counted from the beginning of the window and determines the starting point of the data that is to be transferred. Hence, by setting offset and size appropriately, any portion of the window can be transferred.

In our code, we open the following windows:

- one window for the relocated electromagnetic field of the whole domain: `window_rel_EB`
- three windows comprising the x, y and z current components of the whole domain: `window_j(1:3)`
- one window per slab for the particles: `window_part(1:no_slabs)`

Each of the windows covers a *continuous* region of memory, and, in their simplest form, the `PUT` and `GET` routines transfer continuous portions of these windows. This has to be taken into account when deciding about the direction in which the task slabs are introduced: FORTRAN saves arrays column-wise rather than row-wise. Therefore, the slab `A(1:10,1:10,4:5)` of the array `A(1:10,1:10,1:10)` lies continuously in memory, whereas e. g. `A(4:5,1:10,1:10)` does not. Such slabs of the relocated electromagnetic field array and of the current array have to be transferred via `PUT` and `GET` when performing a task of a remote PE. In order to make sure that the field data within a slab lies continuously in memory, we therefore chose the z direction for introducing the slabs. As they already involve a division of the simulation box in the z direction, we now fix the number of PEs in this direction at 1 and allow the decomposition of the domain only in the x and y directions.

Unlike their MPI-1 counterparts `SEND` and `RECV`, the `PUT` and `GET` routines do not involve the target PE. Hence, the situation might arise that two PEs access the same part of a third PE's memory at the same time. Imagine, for instance, that PE1 is writing into the memory of PE3 while PE2 is reading from it. This would result in PE2 reading partly new and partly old data. In order to avoid numerical errors arising through such concurrent access to remote memory, MPI-2 provides routines to temporarily *lock* a certain window for exclusive access.

We make use of such a lock / unlock routine for every `PUT` and `GET` operation, although the way in which the taskfarm is managed makes sure that no two PEs operate on the same slab, i. e. on the same portion of a PE's memory. Nevertheless, the locking of windows is necessary for two reasons:

1. The `PUT` and `GET` commands of MPI-2 are non-blocking, i. e. one cannot be sure that the data has actually arrived when these routines return! If, however, they are embedded into a lock / unlock pair, the unlock procedure returns only once the data transfer is complete [MPI-2, 1997; Minty, 1998].

2. The current contribution of a certain slab "spills over" at the slab boundaries because of the finite particle size. Upon adding this current to the appropriate portion of the target PE's current array, an overlap with neighbouring slabs occurs. In order to avoid conflicts with other PEs that might concurrently access these neighbouring slabs, the current window has to be properly locked and unlocked for exclusive use.

Hence, the pseudo-code of the routine `do_task` looks as follows:

```
! unpack work package received from master:
target_PE = work_package(1)
slab = work_package(2)
    ...
no_part(1:2) = work_package(10:11)
old_no_part(1:2) = work_package(12:13)

IF target_PE = myid THEN
   ! particle push on old, and current calculation on new number of particles:
   CALL push(slab,old_no_part)
   CALL calc_current(slab,no_part)
ELSE
   ! get relocated fields:
   WIN_LOCK window_rel_EB on target_PE
   GET rel_EB from (target_PE,window_rel_EB,offset,size)
   WIN_UNLOCK window_rel_EB on target_PE
   ! get particles:
   WIN_LOCK window_part(slab) on target_PE
   GET particles from (target_PE,window_part(slab),offset,size)
   WIN_UNLOCK window_part(slab) on target_PE
   ! particle push on old, and current calculation on new number of particles:
   CALL push(slab,old_no_part)
   CALL calc_current(slab,no_part)
   ! add x component of current on target PE:
   WIN_LOCK window_j(1) on target_PE
   ACCUMULATE current on (target_PE,window_j(1),offset,size,MPI_SUM)
   WIN_UNLOCK window_j(1) on target_PE
   ! add y component of current on target PE:
   ...
   ! add z component of current on target PE:
   ...
```

```
   ! put particles back:
   WIN_LOCK window_part(slab) on target_PE
   PUT particles into (target_PE,window_part(slab),offset,size)
   WIN_UNLOCK window_part(slab) on target_PE
ENDIF
```

The MPI_ACCUMULATE routine that appears here is similar to MPI_PUT. It does, however, not *replace* the data of the target PE with the transferred data, but allows to *combine* the two data sets via a variety of arithmetic or logical operations (depending on the data type, [MPI, 1994]). In our case, we use the operation MPI_SUM, which adds the transferred current to the one of the target PE.

The offsets and sizes to be specified in all the PUT and GET operations determine the portion of the respective window that is to be transferred. Since for the particles, each slab has its own window, these parameters are simply `offset=0` and `size=no_part(1)+no_part(2)`. For the transfer of field and current data, the offsets and sizes can be computed from the information given in the work package, which includes the boundaries of the slab in question (see above).

3.2.3 Performance of the taskfarm

The objective of the taskfarm was to distribute the workload on the particles more uniformly among the various PEs. In order to check in how far the implemented taskfarm accomplishes this objective, we first compare a reference simulation run of the optimized code with a similar run of the straightforward, "old" code. For a configuration with $16\times1\times1$ PEs, we recorded for every PE the number of particles being processed in the course of the simulation. The results for the old and new parallel code are shown in the top and bottom panels of Fig. 3.11, respectively.

The particle beam emerges from one side of the simulation box and gradually moves to the other, until it finally extends over the whole grid in the x direction (Fig. 3.12). Whenever the beam front reaches a certain PE, the number of particles in this PE rises linearly. It saturates once the beam front has passed over to the next PE. As stated earlier and now confirmed in Fig. 3.11, this results in a very inhomogeneous distribution of particles: The first PE on the left holds its share of the beam – approximately 1/16 of the final number of particles – practically throughout the simulation, whereas the outermost PE on the right does not receive any particles until the very end.

As expected, the distribution of the particle workload in the optimized version of the parallel code is significantly more homogeneous (bottom panel of Fig. 3.11). The trading of particle groups among different PEs results in a much smaller "bandwidth" of particle numbers. We recall here that the actual distribution of *particles* is still the same as in the old code; it is the particle *work*, i. e. push and

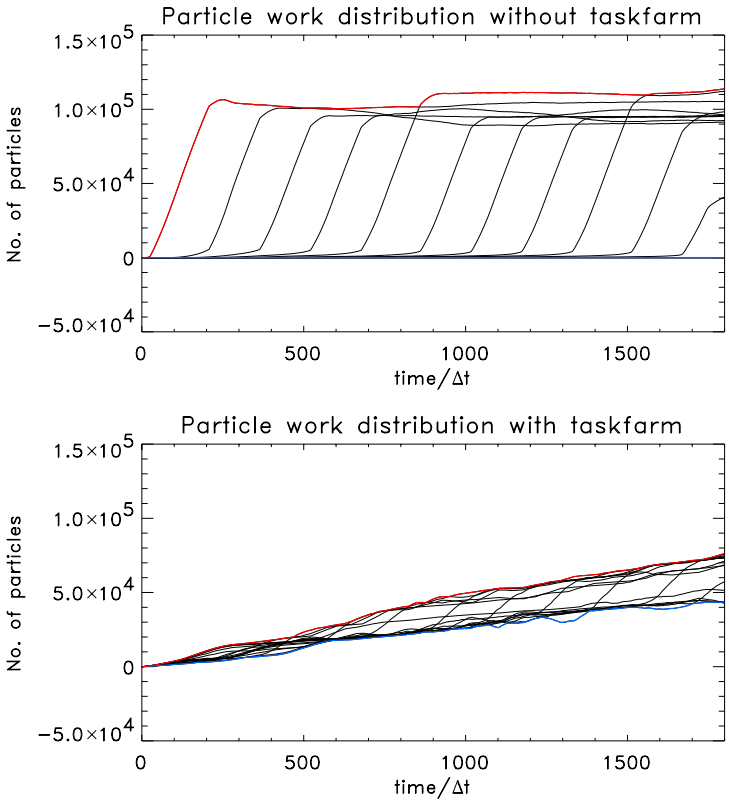

Figure 3.11: The distribution of particle work for a 16×1×1 reference simulation run of the old code (top panel) and the new code (bottom panel). Each trace corresponds to one PE. The red and blue lines mark the maximum and minimum number of particles. Their distance is a measure of the workload imbalance. (For a better readability, the curves in the bottom panel were smoothed by averaging over 50 time steps).

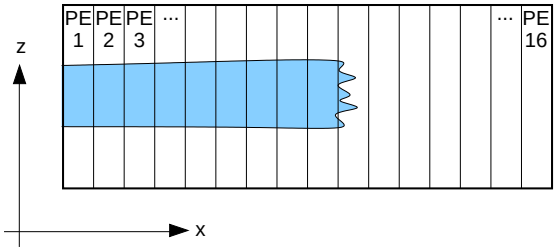

Figure 3.12: The configuration for the 16-PE reference runs. The beam emerges from the left and moves gradually to the right. When it almost reaches the right boundary, the simulation terminates.

current calculation, that is distributed more uniformly by virtue of the taskfarm and that is represented by the lines in the bottom panel of Fig. 3.11.

Since it is the slowest PE that determines the overall speed of the code, the time-averaged maximum particle number should be a good indicator for the total simulation time. For the old and the new code, this number is around 10^5 and $10 \cdot 10^4$, respectively (Fig. 3.11). Judging from these numbers, one would therefore expect a ratio of 2.5 between the simulation times of the new and the old code. Indeed, for the 16-PE reference runs, the actual ratio turned out to be very close to the theoretical value: $T_{new} = T_{old}/2.65$.

The number of slabs `no_slabs` into which the PE domains are subdivided in the z direction was set to 20, because for this number, the acceleration as against the old code was found to be maximal. How the total time of the reference simulation with 16×1×1 PEs depends on `no_slabs` is shown in Fig. 3.13 for slab numbers from 1 to 60, corresponding to slab thicknesses between 60 and 1 grid cells. The simulation time drops steeply when increasing the number of slabs from 1 to 5, then decreases moderately until `no_slabs=20`, and slightly increases for slab numbers beyond 20.

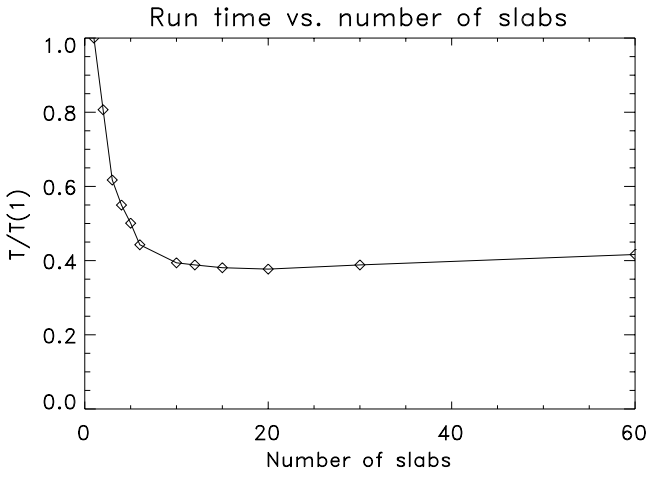

Figure 3.13: The variation of the total simulation time with the number of slabs. The run times are for the 16-PE reference run and are normalized to that of the old code, which corresponds to no_slabs=1.

It is difficult to assess why 20 turned out to be the optimum number of slabs in this particular case. In general, when planning a simulation run, one faces the problem of guessing the right value of `no_slabs` that yields optimum performance. Of course, for the taskfarm to be able to distribute the particle workload equally among all PEs even in situations where all particles reside only on one PE, the number of tasks should be at least of the order of the total number of PEs. A further increase in `no_slabs` results in a finer partitioning of the total work and makes a well-balanced distribution more probable to achieve. How-

ever, with an increasing number of slabs, the size of the tasks becomes smaller, and thus the *overhead* of the taskfarm becomes more important in comparison with the actual work to be performed on the tasks. According to Fig. 3.13, the variation of run time with `no_slabs` beyond the optimum value is only minor. Therefore, by choosing `no_slabs` always close to the maximum possible value (i. e. corresponding to a slab thickness of one grid cell), one can be quite sure not to be too far off from the optimum performance.

Fig. 3.14 displays the profile of the 16-PE reference simulation run. The times shown are again the sum over all PEs including the master, and are categorized into computational routines, taskfarm associated work and communication outside the taskfarm. The latter two each make up roughly 25% of the total simulation time, the remaining 50% being spent on "real" computations.

As far as the time distribution within the computational part of the program is concerned, the particle-based routines are still the most time consuming ones. However, they do not dominate over the grid-based routines as much as they did in the old code (cf. Fig. 3.7). This is not only the case in the 16-PE run, but was observed for every reference simulation. While the computational cost of the grid-based routines stayed roughly the same, the particle push and the current calculation of the new code were found to be quicker than those of the old code by a factor of up to 6!

The *total* particle workload is still the same as in the old code – even though it is now better distributed among the PEs, and the times in Fig. 3.14 are the *sums* over all PEs. Hence, at first sight, the considerable acceleration of the particle routines surprises. It is, however, the smallness of the particle slabs, into which the total particle work is partitioned in the new code, that allows to process the particles much faster: Gathering the particle forces from the grid or scattering the current onto the cells does not require memory accesses to the *whole* PE domain anymore, but only to a considerably smaller part of it. Such a "localization" of data can lead to significant time savings [Anderson et al., 1997; Schüle, 2000], as it does in our simulation, and is a nice *side-effect* of implementing the taskfarm.

The *intended* effect of the taskfarm was to reduce the useless waiting times of those PEs with only a few particles. As can be seen from the last three bars in the bottom of Fig. 3.14, this has been accomplished to a certain degree. The time spent on the receiving operations has indeed decreased to less than a tenth of the total simulation time. The call of `MPI_ALL_REDUCE`, where all PEs synchronize before entering the taskfarm management routine, is, however, a very costly one. The PEs spent 15% of their time here, just waiting for the others to catch up. Altogether, the waiting times outside the taskfarm sum up to a contribution of 25% to the total simulation time. Hence, although being already much better balanced than the old code, the optimized version still exhibits an appreciable imbalance of workload *outside* the taskfarm. Its origin will be discussed further down.

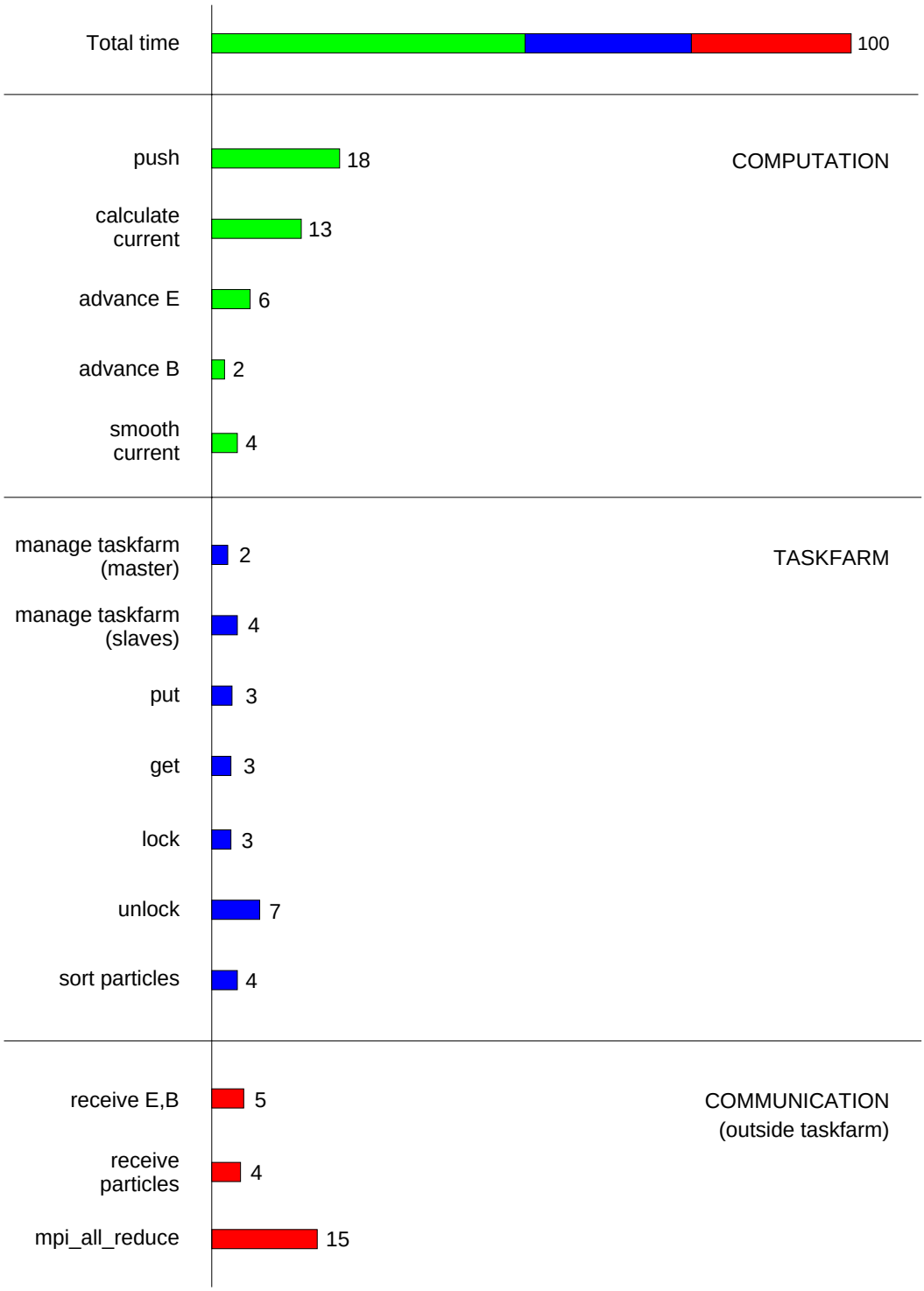

Figure 3.14: The profile of the 16-PE taskfarm reference run. The times shown are the sum over all PEs, including the master. Computational routines account for 50% of the total simulation time, while the taskfarm and the communication outside the taskfarm each contribute a quarter. (Not all routines are shown.)

The time savings thanks to the taskfarm have to be partly paid with additional inter-processor communication and some administrative overhead:

- Taskfarm management: In the taskfarm management routine, the master PE waits for incoming messages from the slaves, which themselves wait for orders from the master (accounts for 2+4=6% of the total simulation time).
- Data transfer: For tasks on remote PEs, particles, relocated field and current have to be fetched and written back (via PUT and GET, 6%). The time spent on locking and unlocking the appropriate windows (10%) is mainly waiting time: Before locking a window for exclusive use, a PE has to wait until no other PE accesses it; and the unlock operation does not return before all the data to be transferred has actually arrived.
- Sorting particles: Grouping all particles into a series of slabs – the essential prerequisite of the taskfarm – requires to sort the incoming particles into their respective slab and to redistribute them after each particle push (4%).

In total, the taskfarm itself accounts for a quarter of the overall simulation time. This might seem much, but, as can be seen from Fig. 3.15, which compares the parallel efficiency of the new code with that of the straightforward parallelization,

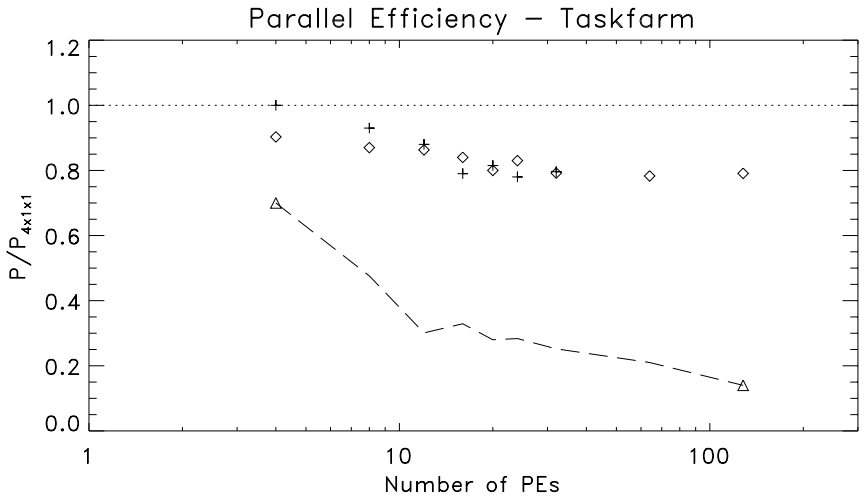

Figure 3.15: The parallel efficiency of the new code for one-dimensional PE-arrangements (+) and such with more than one PE in the y direction (◇). In general, the actual PE arrangement was found not to have a big impact on the efficiency. The dashed line bounded by the triangles shows the maximum achievable efficiency of the old code (cf. Fig. 3.6). The efficiencies are all normalized to that of the $4\times1\times1$ run of the new code, and the number of PEs does not include the master (no_slabs=60 for PE numbers greater than 32, for all other runs no_slabs=20).

it is worth it: While the efficiency of the old code decreases continuously to a value of 0.15 for 128 PEs (normalized to the $4\times1\times1$ run of the new code), the taskfarm's efficiency stays essentially constant at 0.8 for PE numbers beyond 20. Hence, the acceleration as against the old code rises for increasing PE numbers and reaches a factor of more than 5 for 128 PEs.

3.3 Assessment of parallelization strategies

In the preceding two sections, we developed two MPI-based parallel versions of our simulation code. One of them employs the standard parallelization technique for plasma PIC simulations: a static, equidistant domain decomposition in up to three dimensions. It was found to run fairly efficiently on up to 8 processors. For larger PE numbers, however, the speed-up drops to an unacceptable level (Fig. 3.6).

As a reason for the poor efficiency of the straightforwardly parallelized code we identified the strong inhomogeneity in the particle distribution: In our simulations of ion thrusters, the particles are injected into the simulation box from one side and gradually move to the right. This causes the share of the particle beam to vary considerably from domain to domain, and thus involves a large imbalance of the particle associated workload.

In the optimized version of our parallel code, we reduced this workload imbalance by incorporating a taskfarm-based dynamic load-balancing scheme – a strategy that has not been used in plasma PIC simulations so far. It allows PEs with less particles to take over some of the particle work of other, busier PEs and thus helps to equalize the distribution of the particle associated workload among all PEs. For our code, the taskfarm yielded a parallel efficiency of about 0.8 for up to 128 processors, which corresponds to an acceleration by a factor of more than 5 as against the straightforward parallelization.

This acceleration is only partly due to the more uniform distribution of particle workload. As a side-effect of partitioning the workload into smaller tasks, a stronger localization of the data is achieved, which leads to significant savings in the particle routines themselves.

Thanks to the taskfarm, the two particle-based routines – push and current calculation – are now quite well-balanced. However, as manifested by the cost of the synchronization call to `MPI_ALL_REDUCE` (Fig. 3.14), there is still some degree of imbalance outside the taskfarm. Since the equidistant decomposition was kept, this cannot arise through the grid routines. What actually causes the imbalance is the "administration" of the particles: They have to be sorted into their slabs (cf. Fig. 3.14) and have to be packed into / unpacked from buffers in the particle exchange routine (see Sec. 3.1.2). All this involves loops over all particles and is thus proportional to their total number. The taskfarm in its present form only

balances the *computational work* on the particles, but leaves their distribution as it was in the old code. Hence, the remaining imbalance could be reduced by also "out-sourcing" the particle administration: Apart from the particle push and the current calculation, each task would then also comprise their sorting and packing / unpacking.

This is a possible modification of the taskfarm, which is quite likely to further enhance its efficiency. In its present form, however, the taskfarm constitutes already a major improvement to the straightforward parallelization scheme. The code is now well suited to run on a massively parallel computer and was successfully employed to obtain the simulation results of the following chapter.

Chapter 4

Ion thruster beam neutralization

The process of ion thruster beam neutralization, i. e. the mixing of electrons and ions, presents a challenging problem: In order to neutralize both the ion charge density and the ion current, the electrons being emitted with random velocities from a small cathode on one side of the ion beam have to spread out across the whole beam cross section, and also have to adapt their average velocity to the ion bulk movement. Due to the essentially collisionless character of the beam plasma, this adaptation cannot simply rely on particle-collisions.

So far, the physics that govern the neutralization process have not been studied in detail. As stated in the introduction, some fundamental questions are yet to be dealt with. These questions address the mechanism of thermalization, possible plasma instabilities occurring upon neutralization and the impact of basic beam parameters such as injection velocity ratio, beam width and ambient magnetic field.

Aiming at understanding those issues, we now apply our simulation code to the problem of ion thruster beam neutralization. As a first approach to this complex scenario, we start our investigations in a simplified geometric configuration by disregarding the spatial separation between the ion source and the electron emitting cathode (Sec. 4.1). Both species are thus injected through the same rectangular opening on the left-hand side of the simulation box, resulting in a quasi-1D geometry with major variations in the axial direction (Fig. 4.1).

In Sec. 4.2, we investigate the effects of a static magnetic field, as generated e. g. by permanent magnets in the thruster chamber, on the quasi-1D neutralization process. The dependence of the downstream plasma state on magnetic field strength and beam width is studied in detail.

Our simulation results for a fully three-dimensional configuration with spatially separated electron and ion sources are presented in Sec. 4.3, where we characterize the neutralization process for a variety of injection velocity ratios. On the basis of our findings, we finally analyse some of the measurements of the ion thruster environment on Deep Space 1.

4.1 Neutralization in a quasi-1D configuration

We start our investigations by simulating the region downstream of the thruster exit in the quasi-1D geometry of Fig. 4.1. Equal numbers of electrons and ions are injected at each time step. As the densities of the charge-exchange ions (CEX) and of the ambient solar wind plasma are smaller than the ion beam density by a factor of 4×10^3 [Wang et al., 2000] and 4×10^9, respectively, these two plasma populations do not have a significant impact on the neutralization and are therefore neglected in our simulation. At the present stage of our investigations, we also neglect the static magnetic field that arises from the permanent magnets in the thruster chamber of DS1 [Brinza et al., 2000].

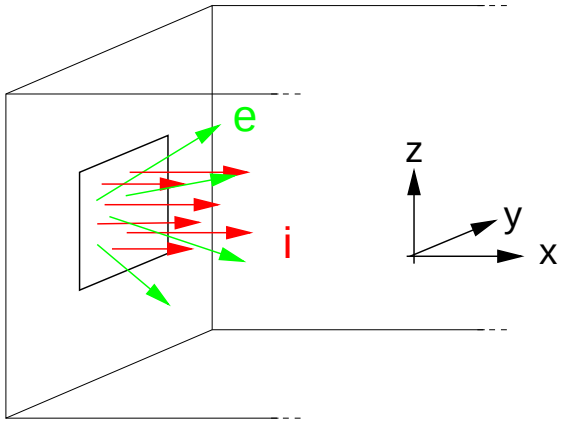

Figure 4.1: The simulation geometry. Electrons and ions are injected randomly distributed over a square on the left-hand side of the simulation box. Major variations are expected to occur in the x direction.

We do not specifically simulate the DS1 thruster, but for the simulation parameters, the actual DS1 values, some of which are displayed in Tabel 1 (Ch. 2), served us as a rough guidance. Our code being fully electromagnetic requires, however, some rescaling of the actual particle velocities towards higher values. As it is not the absolute values of the involved velocities but rather their ratio, this should not alter the physics significantly. Hence, we are free to fix one velocity at a certain value, which we can choose according to numerical constraints, and can simulate different scenarios by scaling other velocities appropriately.

The thermal velocity of the electrons $v_{e0}^{th} = \sqrt{k_B T_{e0}/m_e}$, which are injected with a half-Maxwellian velocity distribution in v_x and a full-Maxwellian in v_y and v_z, is therefore fixed at $0.1c$ throughout this work, where c is the velocity of light. As v_{e0}^{th} also affects the electron Debye length and thus the grid spacing (see Sec. 2.7), a constant value of v_{e0}^{th} allows a proper comparison between different simulation runs. Ions leave the injection plane with a bulk velocity v_{i0} plus a small thermal spread corresponding to $T_i = T_{e0}$.

The grid spacing ΔX is set equal to the electron Debye length λ_D, and the time step is chosen such that $\Delta t = 0.4\Delta X/c$. The electron plasma frequency in the fully neutralized beam, i. e. for $n_e = n_{i0}$, is $\omega_{pe}^0 \Delta t = 0.027$, leading to an electron inertia length l_e of $l_e = c/\omega_{pe}^0 = 14\Delta X$.

We simulate beam diameters between 0.36 and $3.6 l_e$ or 5 and $50\lambda_D$. These diameters are comparable to the DS1 value when measured in l_e (Table 1). However, as measured in λ_D, we actually simulate a scaled-down version of an ion thruster. In order to be applied to actual ion engines, our results have to be rescaled appropriately, e. g. by matching ratios of characteristic lengths, times or velocities [Wang et al., 1996].

Due to the actual electron to ion mass ratio of roughly 1:250,000, the time scales of the respective particle species are widely separated. In order to cover both time scales within the simulation run time, this mass ratio could be increased. However, throughout this work, we rather use the actual mass ratio and focus solely on the high-frequency electron dynamics. We run the simulations typically for about $\omega_{pe}^0 t = 600$, which corresponds to 0.2 ion plasma periods. Therefore, in our simulations the ion dynamics do not play a role at all, apart from their bulk movement that generates the fields with which the electrons interact.

The initial condition of our simulations is a field-free vacuum, and the boundary conditions are as described in Sec. 2.5. Unless stated otherwise, electric field values and densities are averaged across the beam cross section for diagnostics, thus giving a very good signal-to-noise ratio.

4.1.1 Electron dynamics for different velocity ratios η

This section is dedicated to exploring the impact of the injection velocity ratio $\eta = v_{e0}^{th}/v_{i0}$ on ion thruster beam neutralization. We therefore fix the beam width b for the simulation runs of this section at $10\lambda_D$. The velocity ratio η is varied between 1.0 and 2.0, because it turned out to be this range where the plasma makes a remarkable transition between two fundamentally different behaviours. The results for the case $\eta = 1$ are presented in Figs. 4.2–4.5.

Fig. 4.2 shows the velocity distribution of the ions and their density in the axial direction after $\omega_{pe}^0 t = 200$. The velocity is normalized to v_{i0}, and the density to the nominal ion density of a non-diverging beam n_{i0}. This normalization is adopted throughout this work, unless stated otherwise. By this time, the ion beam has expanded from the injection plane at $x = 10\lambda_D$ up to $x = 270\lambda_D$[1]. Due to the huge mass of the ions, their velocity and spatial distribution are essentially unaffected and are equal to what would be expected from free flow. The ion beam shows only a minor divergence, as can be seen from the slight decrease in density. Apart

[1]Unless stated otherwise, positions and distances are always given in units of λ_D. Where not misleading, we therefore omit "λ_D" from now on.

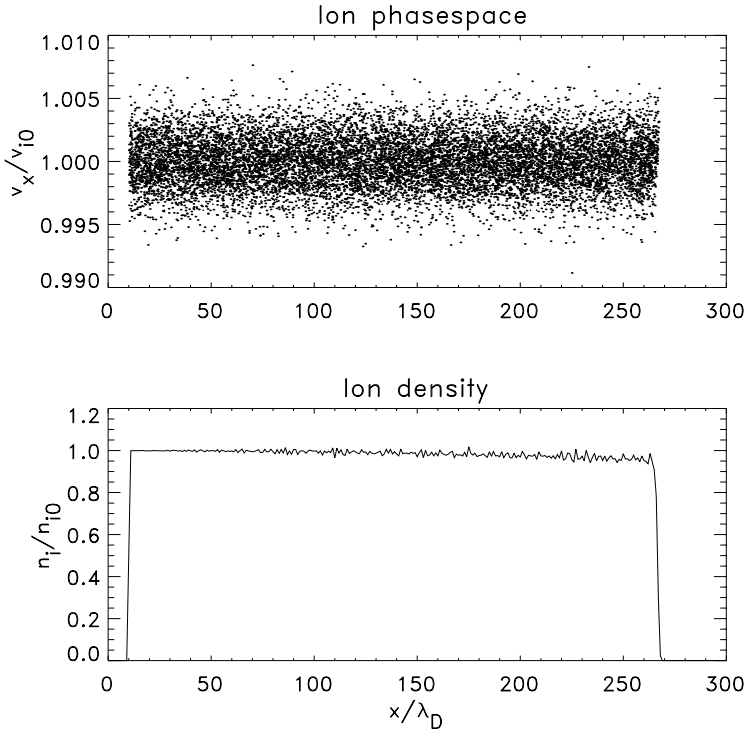

Figure 4.2: Ion phase space and ion density for $\eta = 1$ and $\omega_{pe}^0 t = 200$.

from Sec. 4.3, this holds for all the simulation runs of this work. Thus, the ions just form the expanding beam and its associated potential but are otherwise irrelevant for the plasma processes to be presented.

The electric potential, the electron phase space, and their density for $\omega_{pe}^0 t = 200$ are shown in Fig. 4.3. The potential was obtained by averaging the axial component of the electric field E_x across the beam, integrating in axial direction and setting $\Phi = 0$ in the injection plane ($x = 10$). It is normalized to $\Phi_0 := k_B T_{e0}/e$. As can be seen from Fig. 4.3, the plasma beam breaks down into five regions:

1. "Acceleration region": A sheath of about $10\lambda_D$ thickness adjacent to the injection plane, where the potential rises steeply to about $3\Phi_0$. In this sheath, the injected electrons are accelerated to about twice the ion bulk velocity.

2. "Beam region": A wavy high potential region with $\Phi \approx 3\Phi_0$. The electrons in this region are roughly equipartitioned between a strong beam at $v \approx 2v_{i0}$ and a trapped hot component (cf. Fig. 4.4).

3. A "thermalization region" of 10–$15\lambda_D$ thickness, where the potential drops by $2.3\Phi_0$, and the electron density jumps by a few percent of n_{i0}.

4. A "quiescent region" of constant potential at $\Phi = 0.75\Phi_0$. Noting that the majority of electrons are injected with $v_e = 0$, this value of Φ corresponds

roughly to the energy that is necessary to accelerate them to v_{i0}. The plasma in this region is thermalized: The electron distribution function is a drifting Maxwellian with a drift velocity equal to the ion bulk velocity v_{i0} and a temperature $T_e = 0.32T_{e0}$. The reason for T_e being less than the injection value is the escape of fast electrons from the attractive potential of the ions. We note that the electron density is only at about $0.7n_{i0}$, i. e. the plasma is highly non-neutral.

5. The "vacuum region" ahead of the ion beam with a strongly decreasing electron density and a potential that falls down to an asymptotic value of $-3.8\Phi_0$.

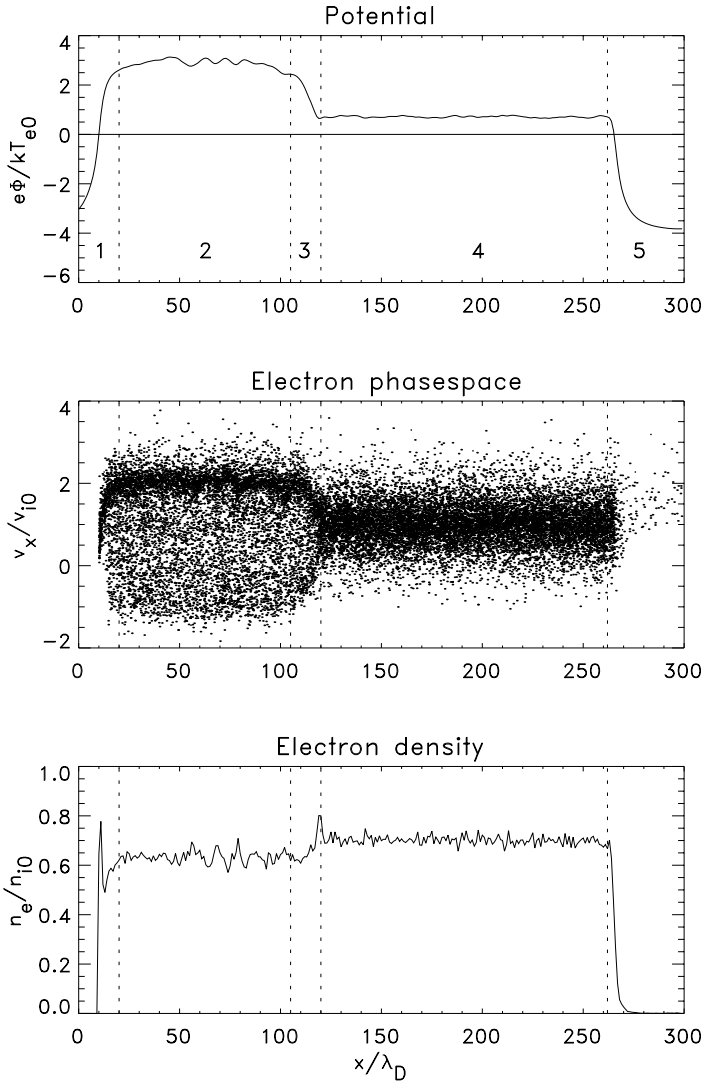

Figure 4.3: Potential, electron phase space, and electron density for $\eta = 1$ and $\omega_{pe}^0 t = 200$. Note the five distinct regions of the plasma beam.

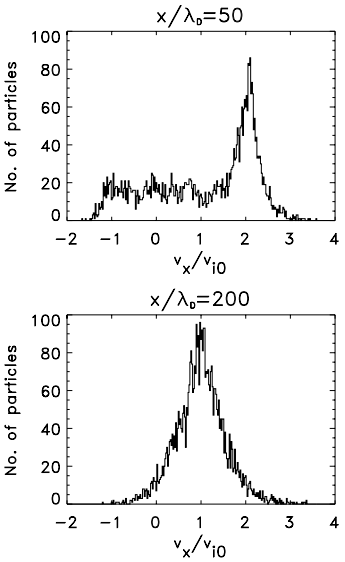

Figure 4.4: Electron velocity distribution function upstream ($x = 50\lambda_D$, upper panel) and downstream ($x = 200\lambda_D$, lower panel) of the shock front for $\eta = 1$ and $\omega_{pe}^0 t = 200$.

The temporal development of these regions from $t = 0$ to $\omega_{pe}^0 t = 400$ is presented in Fig. 4.5, where contours of the axial electric field E_x are shown. As can be seen in this Figure, the acceleration sheath adjacent to the injection plane (region 1) is time stable. Such sheaths are a well-known phenomenon from plasma emission problems [Chen, 1974, p. 290].

The intense wave activity in region 2 can be attributed to beam Langmuir oscillations: Taking $n_e^{beam} \approx 0.5 n_{i0}$ from Fig. 4.4 for the beam density, one gets $\omega_{pe}^{beam} \approx 0.7\omega_{pe}^0$. This agrees well with the frequency of $0.75\omega_{pe}^0$ as obtained from counting the wave maxima at a fixed x in Fig. 4.5. The beam Langmuir oscillations are presumably due to a beam-plasma instability that is driven by the beam passage through the hot background plasma.

The waves propagate towards the potential jump (region 3), which itself is time stable and moves at a velocity of $v_{sh} = 0.38 v_{i0}$. At this potential jump, the electrons are decelerated. Those with sufficient kinetic energy emerge on the downstream side (region 4) to form the thermalized plasma. The others are reflected and are thus trapped in the expanding high potential region between the injection sheath and the moving potential jump. They constitute the hot plasma component in region 2.

The region around the potential jump that separates two distinct plasma regimes, one of which is a thermalized plasma, on a scale of only about one electron inertia length, qualifies itself as a moving electrostatic shock. It is by virtue of this shock that the plasma, which is far away from thermal equilibrium after injection, thermalizes. The questions arising now are: (a) how is the shock formed and (b) what determines its velocity.

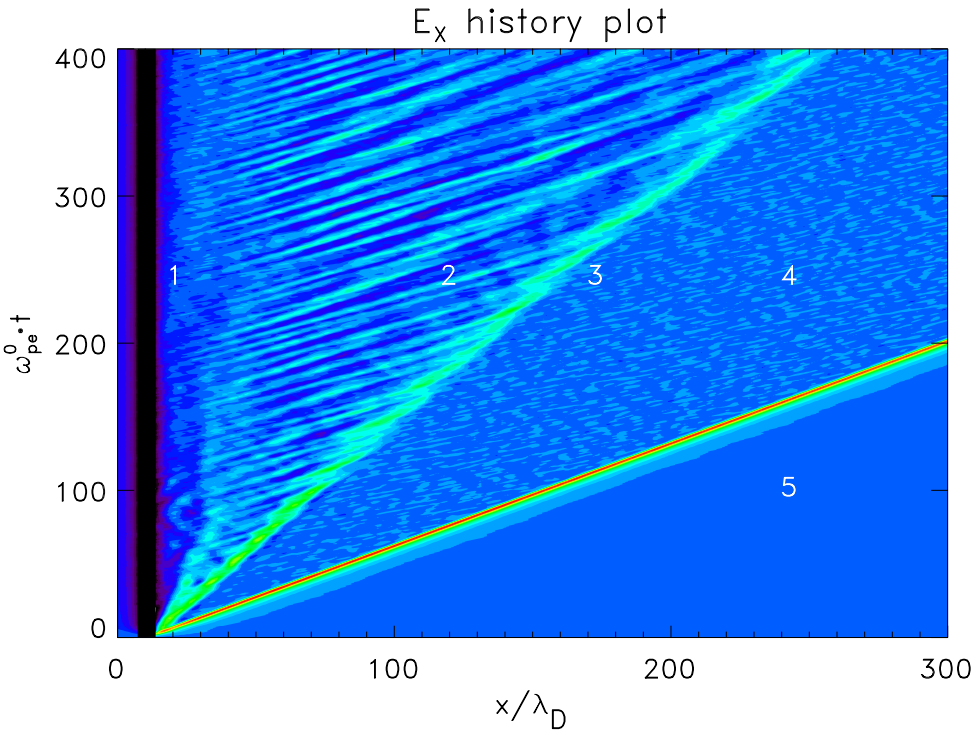

Figure 4.5: History plot of the axial electric field E_x for $\eta = 1$. The different regions are numbered as in Fig. 4.3. Note the acceleration region (1), the beam-plasma unstable waves (2), the shock front (3), the quiescent "lake" of thermalized plasma (4), and the vacuum in front of the ion beam (5). The blue tone in region 5 corresponds to zero electric field, and "hotter" and "colder" colours to positive and negative E_x, respectively.

While the latter will be investigated in the next section, the formation of such a shock can be understood by looking at a special family of electrostatic shocks, namely double layers. Such structures require at least three different particle populations: cold streaming electrons, cold streaming ions, and trapped electrons [Hudson and Potter, 1981], all of which are present here. One possible mechanism for the formation of double layers was suggested by Hasegawa and Sato [1982]: Drifting electrons are reflected by a negative charge density spike that is thought to be due to the trapping of electrons and ions in the potential of ion acoustic waves. The reflected electrons enhance the negative charge density at that point, and leave a positive charge excess on the downstream side, thereby leading to the double layer potential profile. The "double layer" in our simulation arises in a similar manner with the required reflection of electrons taking place at the leading edge of the expanding ion beam.

However, there is a fundamental difference between common double layers and the structure in our simulation: Double layers manifest themselves in the distribution functions of electrons as well as of ions. Both species show a density

perturbation across the double layer, so that the plasma is quasi-neutral on both sides. The structure in our simulation, though, arises only from the electron component, while the ion distribution function is practically unchanged across the shock front. Even though on both sides the plasma departs from quasi-neutrality, the respective potentials are constant. In 1D such a configuration is impossible since $d^2\Phi/dx^2 = e(n_e - n_i)$. Hence, a proper understanding of this potential jump and of ion beam neutralization in general requires at least a two-dimensional approach, which allows for electric flux leaving through the lateral surfaces of the beam. The usually one-dimensional results obtained for double layers cannot be directly applied to our shock.

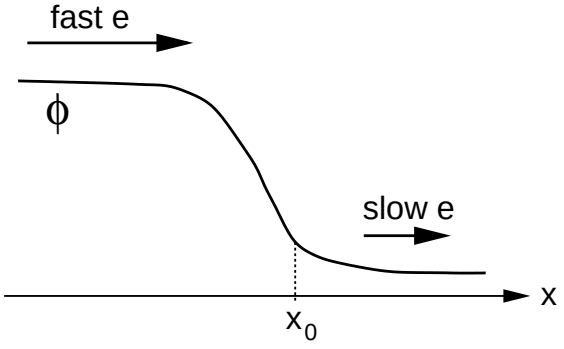

Figure 4.6: 1D potential profile to derive Eqn. (4.6).

A *necessary* condition for the observed potential profile to build up can be derived quite easily, and goes back to Sagdeev [1979]. We consider only the streaming electrons on both sides of the shock (Fig. 4.6), and combine the electron equation of motion

$$n_e m_e v_e \frac{dv_e}{dx} = n_e e \frac{d\Phi}{dx} - \frac{d}{dx}(n_e k_B T_e) \tag{4.1}$$

with the electron continuity equation

$$\frac{d}{dx}(n_e v_e) = 0 \tag{4.2}$$

to obtain

$$\frac{1}{v_e}\frac{dv_e}{dx}(m_e v_e^2 - k_B T_e) = e\frac{d\Phi}{dx} \quad , \tag{4.3}$$

which is now evaluated on the downstream side at x_0 in Fig. 4.6. The electrons are assumed to be drifting to the right and to be slowed down, i. e. $v_e > 0$ and $dv_e/dx < 0$, while $d\Phi/dx < 0$. Hence, a necessary condition for a structure as depicted in Fig. 4.6 to be compatible with Eqn. (4.3) is $m_e v_e^2 - k_B T_e > 0$, or

$$v_e > v_e^{th} \tag{4.4}$$

on the downstream side. In the case of an ion thruster, the downstream side is characterized by a thermalized plasma, i. e. the electron drift velocity equals the ion beam velocity v_i. Thus, the necessary condition reads

$$v_i > v_e^{th}, \tag{4.5}$$

or, as $v_i \equiv v_{i0}$ holds in our case,

$$v_{i0} > v_e^{th}. \tag{4.6}$$

That is, the ion beam velocity has to be greater than the electron thermal velocity on the downstream side in order for the observed potential jump to develop. The downstream temperature in the presented simulation run with $\eta = v_{e0}^{th}/v_{i0} = 1$ is $T_e = 0.32T_{e0}$, i. e. $v_e^{th} = 0.57v_{i0}$, which fulfills the necessary condition.

It is of interest now what will happen when the necessary condition Eqn. (4.6) is not met. A potential jump should not develop. We verified that by simulating the case $\eta = 2$ ($v_{e0}^{th} = 0.1c$ and $v_{i0} = 0.5v_{e0}^{th}$). As can be seen in Figs. 4.7 and 4.8, which show potential, electron phase space and electron density for $\eta = 2$, the electron dynamics differ significantly from the results for $\eta = 1$, and a potential jump does indeed not develop.

A stable sheath, similar to region 1 of the previous simulation, builds up adjacent to the injection plane, where the potential rises to about $1\Phi_0$, as compared to $3\Phi_0$ for $\eta = 1$. The electrons are accelerated to form a beam at about $2.8v_{i0}$, which is, however, much weaker than for $\eta = 1$. As before, the potential shows oscillations that can probably be attributed to beam-plasma unstable waves.

Further downstream, the electrons adopt a velocity distribution that is, in contrast to the case $\eta = 1$, not Maxwellian, but rather a double-peaked function with one peak around $v = -0.8v_{i0}$ and the other at about $v = 2.6v_{i0}$. While the latter can be identified with the broadened and weakened left-over of the electron beam formed in the acceleration sheath, the former is probably due to the reflection of this left-over at the front end of the ion beam.

The overall electron drift velocity is equal to the ion beam velocity. In this respect, a thermalization of the plasma takes place also for $\eta = 2$; however, in contrast to the case $\eta = 1$, not by virtue of a shock wave, but by some other mechanism. Candidates for this adaptation mechanism are two-stream instability between the electrons drifting in the $+x$-direction and those reflected at the front end of the ion beam, and – as will be discussed in Sec. 4.3.2 – a quasi-Fermi-deceleration between the expanding ends of the ion beam.

In a series of runs with different η between 1.0 and 2.0 we investigated where and how the transition between shock and shock-free solutions occurs. The critical value for η was determined as $\eta_{crit} \approx 1.7$. For $\eta > 1.7$, shock-free solutions develop that look like the one in Fig. 4.7, while for $\eta < 1.7$, a shock front builds up similar to Fig. 4.3. The associated potential jump gets lower and lower when η approaches η_{crit} from below, as can be seen in Fig. 4.9, which shows potential, electron phase space, and electron density for $\eta = 1.66$. The velocity distribution function for this case is depicted in Fig. 4.10. It marks the transition be-

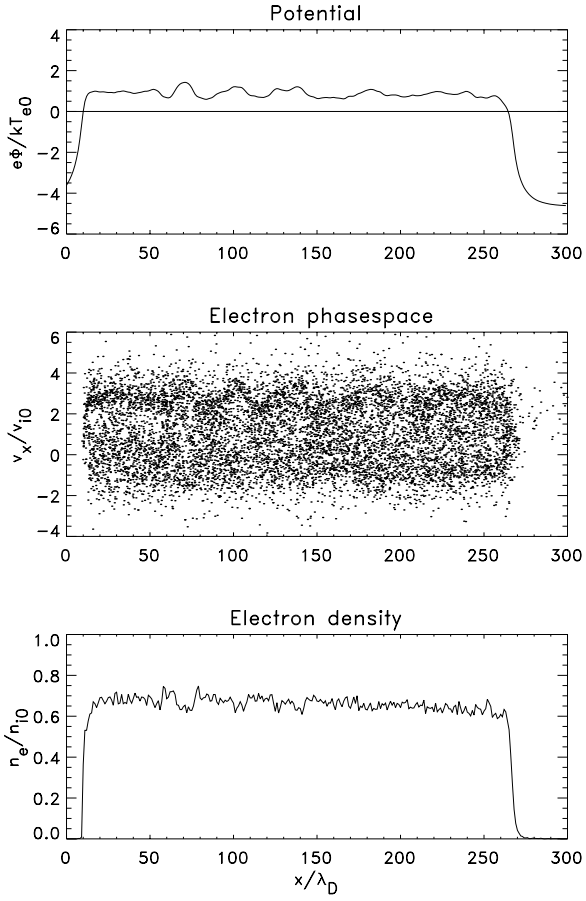

Figure 4.7: Potential, electron phase space and electron density for an injection velocity ratio of $\eta = 2$ and $\omega_{pe}^0 t = 400$.

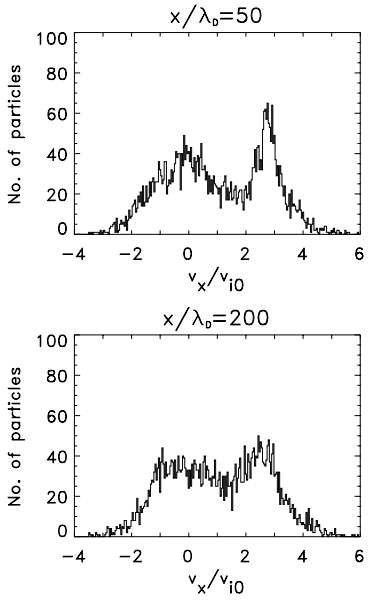

Figure 4.8: Electron velocity distribution function at $x = 50\lambda_D$ (upper panel) and $x = 200\lambda_D$ (lower panel) for the simulation of Fig. 4.7.

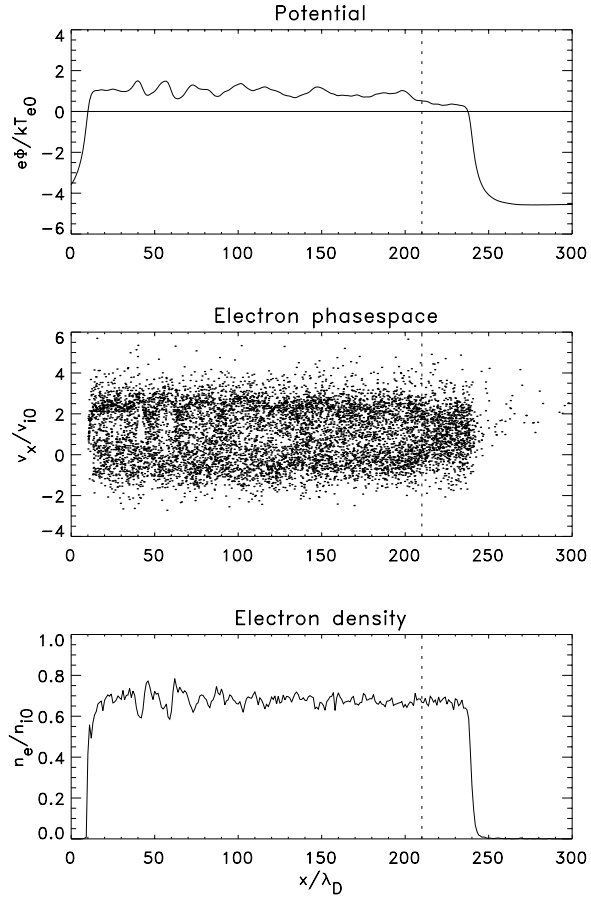

Figure 4.9: Potential, electron phase space, and electron density for $\eta = 1.66$ and $\omega_{pe}^0 t = 300$. Downstream of the dashed line the electrons are thermalized.

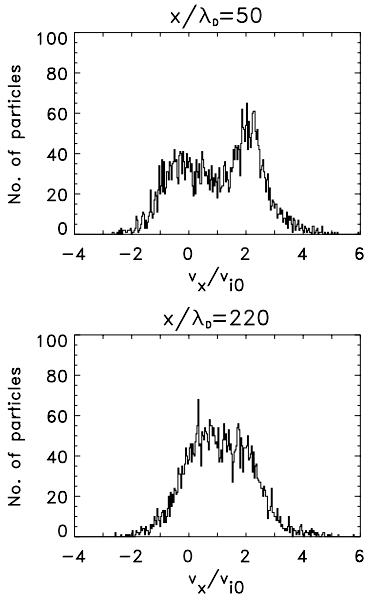

Figure 4.10: Electron velocity distribution function upstream and downstream of the shock front for the simulation of Fig. 4.9.

tween shock solutions and those without shock: The electron beam at $x = 50\lambda_D$ is weaker than that for $\eta = 1$, but still stronger than the beam for $\eta = 2$. The downstream distribution function is much flatter than the Maxwellian of $\eta = 1$, thus indicating the transition to the double-peaked type of distribution function of $\eta = 2$.

The downstream electron temperature for all simulation runs with shock is about $T_e \approx 0.32T_{e0}$, giving a downstream electron thermal velocity of $v_e^{th} \approx 0.57v_{e0}^{th} \approx \eta v_{i0}/1.77$. Hence, our observations are perfectly consistent with the necessary condition Eqn. (4.6) for the formation of a shock front.

4.1.2 Dependence on lateral beam dimension

In order to further investigate the shock wave, we studied its behaviour for varying lateral beam dimensions between $b = 5\lambda_D = 0.36l_e$ and $b = 50\lambda_D = 3.6l_e$. η is fixed at 1.0 with $v_{e0}^{th} = v_{i0} = 0.1c$. The ion injection rate is adjusted for each run to give a constant ion density, and as before, electrons and ions are injected at the same rate.

Fig. 4.11 shows the potential at $\omega_{pe}^0 t = 110$ for $b = 10, 20$ and $40\lambda_D$. In the injection plane we defined $\Phi = 0$, as before. Apparently, the general morphology of the potential is independent of the beam width and looks as described in the previous section. The potential levels in the wavy high-Φ region and in the plateau region behind the shock front are around $3\Phi_0$ and $0.7\Phi_0$ for all b. As the beam is not fully neutralized and the total charge of the ions increases with b^2, the ambient potential level decreases with increasing beam width.

A remarkable difference between the potential profiles for different b, however, is the shock front velocity. As deduced from the different positions of the shock front in Fig. 4.11, the shock velocity ranges from $0.38v_{i0}$ for $b = 10\lambda_D$ to $0.23v_{i0}$ for $40\lambda_D$. A double-logarithmic plot of the normalized shock front velocity v_{sh}/v_{i0} vs. b/λ_D for an extended simulation series is depicted in Fig. 4.12 (upper panel). A linear fit to this curve yields a relation according to

$$v_{sh}/v_{i0} = 0.91 \cdot (b/\lambda_D)^{-0.38}. \tag{4.7}$$

A dependence of wave velocities on lateral dimensions is a well-known phenomenon of waveguides, where the lateral dimensions determine the perpendicular wave numbers and thus affect the dispersion in the axial direction. Indeed, the ion beam constitutes a plasma-filled waveguide, and the question arises whether the variation of the shock front velocity with varying b is just a consequence of the waveguide mode character of the shock wave.

Plasma-filled waveguides were thoroughly investigated by Trivelpiece and Gould [1959]. These authors found that the mode structure of plasma-filled waveguide

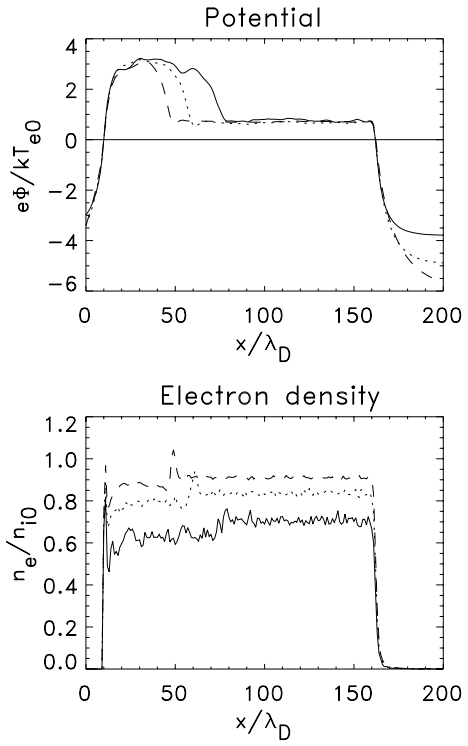

Figure 4.11: Potential and electron density for $b = 10\lambda_D$ (solid line), $20\lambda_D$ (dotted), and $40\lambda_D$ (dashed). Note the different positions of the shock front ($\eta = 1.0$, $\omega_{pe}^0 t = 130$).

modes is such that the axial electric field maximizes at the plasma-vacuum interface and decreases to both sides, with the electron inertia length l_e being the characteristic scale length for the inward decay. In our case, however, the axial electric field E_x is quite constant within the beam, as may be seen from Fig. 4.13, which shows the variation of E_x along a perpendicular direction for $b = 40\lambda_D = 2.85 l_e$. There is no sign of an inward decay on the scale of an electron inertia length l_e. Hence, the shock wave in our simulations is not a Trivelpiece-Gould mode, and the fact that the beam is finite in the lateral directions does not seem to directly affect its dispersion characteristics by determining a lateral wave number.

The actual effect of the beam width b on the shock front velocity is rather an indirect one: An increasing beam width decreases the surface to volume ratio of the ion beam, and thus reduces the fraction of electrons that can escape from the ions through the lateral beam surfaces. This affects the electron density in the beam and results in a varying degree of neutralization, ranging from 70% for $b = 10\lambda_D$ up to about 90% for $b = 40\lambda_D$, as may be seen from Fig. 4.11. While the electron phase space looks like that in Fig. 4.3 for all b, the electron density n_e varies significantly. As n_e affects the wave dispersion fundamentally, the difference in electron density for varying b is likely to be the actual cause for the variation of the shock front velocity.

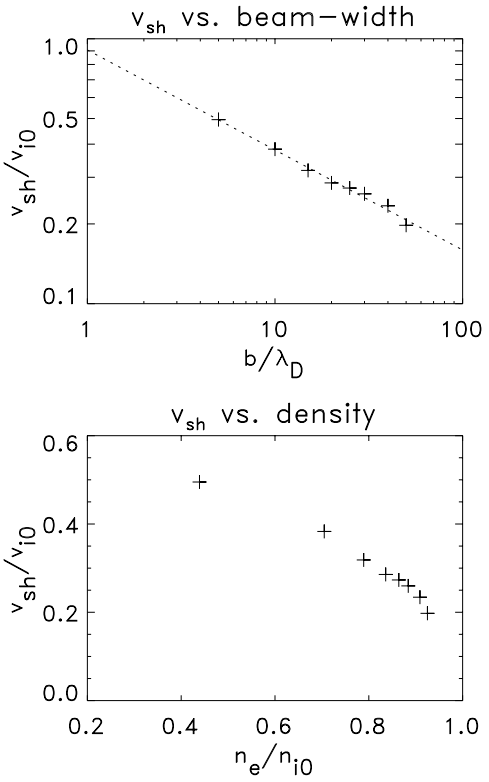

Figure 4.12: Dependence of the shock front velocity on beam width (double-logarithmic plot, upper panel) and on downstream electron density (linear plot, lower panel) for $\eta = 1.0$.

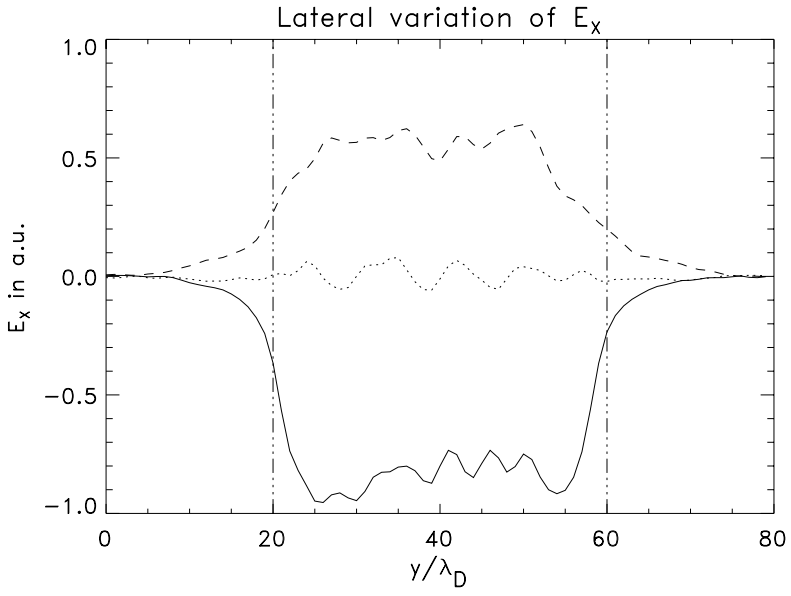

Figure 4.13: The variation of the axial electric field E_x across the beam in the acceleration sheath (solid line), the shock front (dashed) and in the downstream region (dotted). The ion beam extends from $y = 20\lambda_D$ to $y = 60\lambda_D$ ($\eta = 1.0$, $\omega_{pe}^0 t = 130$, $b = 40\lambda_D$).

The lower panel of Fig. 4.12 shows a linear plot of shock front velocity v_{sh}/v_{i0} vs. downstream electron density n_e/n_{i0}. The curve seems to bend down as n_e/n_{i0} approaches unity, indicating that the shock wave is a phenomenon that is tightly coupled to the non-neutrality of the plasma rather than to the absolute electron density. We will come back to this point when investigating beams that are fully neutralized by virtue of an axial magnetic field (Sec. 4.2). In any case, the shock velocity shows a strong dependence on the electron density. Based upon this finding, we will try to qualitatively explain our observations by drawing analogies between an ion thruster beam and an electron diode in the following section.

4.1.3 The thruster beam as an expanding electron diode

In the classical short-circuited electron diode, the electrons enter the diode region with a velocity v_0 through a grid electrode at potential φ_0, and leave it through a second electrode at the same potential. This configuration bears some resemblance to a snapshot of an expanding ion thruster beam, with the injection sheath and the front end of the ion beam corresponding to the respective electrode grids (see Fig. 4.14). Hence, as a first approximation, an ion thruster beam may be regarded as an expanding electron diode. The results obtained for electron diodes, e. g. for the potential profile, can then be transferred to ion thruster beams simply by expanding them self-similarly according to the velocity of the beam.

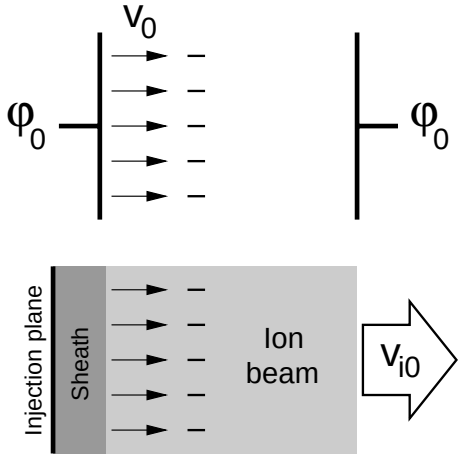

Figure 4.14: Analogies between electron diode and ion thruster beam.

Such a model has of course some shortcomings: (a) electron diodes are treated as 1D devices, whereas the presented results are essentially non-1D phenomena (cf. Sec. 4.1.1), (b) in classical electron diodes there is no neutralizing ion background, as there is in the case of an ion thruster beam. A neutralized electron diode is realized in the Pierce diode [Pierce, 1944]. Nevertheless, for the sake of simplicity, we restrict ourselves to a 1D non-neutralized short-circuited electron diode and attempt to transfer the results for this kind of diodes to ion thruster beams. Despite

its shortcomings, this simple model already reflects some basic characteristics of an ion thruster beam, and can qualitatively explain our observations.

A thorough treatment of electron diodes was carried out, e. g. by Birdsall and Bridges [1966]. In the following, we briefly present their results for a 1D short-circuited electron diode. The governing equations are Poisson's equation, current continuity, and conservation of energy:

$$\frac{d^2\varphi(x)}{dx^2} = -en(x)/\varepsilon_0 \,, \tag{4.8}$$

$$en(x)v(x) = en_0v_0 =: J_0 \,, \tag{4.9}$$

$$\frac{1}{2}mv^2(x) - e\varphi(x) = \frac{1}{2}mv_0^2 - e\varphi_0 = 0 \,. \tag{4.10}$$

They can be combined to give

$$\frac{d^2\varphi(x)}{dx^2} = \frac{J_0}{\varepsilon_0\sqrt{2e\varphi(x)/m}} \,. \tag{4.11}$$

This equation has two well-known types of solutions, depending on the parameter α,

$$\alpha = J_0/J_c \,, \tag{4.12}$$

where J_0 is the electron current density (Eqn. 4.9) and J_c is the Child-Langmuir critical current density for space charge limited current in a plane diode with electrode spacing d:

$$J_c = \frac{4}{9}\sqrt{\frac{2e}{m}}\frac{\varepsilon_0\varphi_0^{3/2}}{d^2} \,. \tag{4.13}$$

For $0 \le \alpha \le 8$, the potential between the electrodes forms a symmetric trough, whose depth increases with rising α (Fig. 4.15). Another type of solution exists for $4 \le \alpha \le \infty$: A so-called *virtual cathode* builds up at x_{vc} between the electrodes, where the potential drops to zero. At x_{vc} the electrons come to a halt and part of them are reflected, while the rest is accelerated away in the $+x$ direction.

The virtual cathode is located exactly in the diode center for $\alpha = 4$, and moves upstream for higher α. The interesting point to note now is the analogy between the virtual cathode solution of an electron diode and the shock solution that we observed in our simulations: In both cases, the potential drops to a value that forces some of the electrons to be reflected. On the upstream side of the virtual cathode or, respectively, of the shock, the electron plasma is composed of reflected electrons and those streaming in $+x$ direction, while on the downstream side, all electrons are streaming toward the exit electrode or the front end of the ion beam, respectively.

Based upon these analogies, the observed shock front may be regarded as a kind of virtual cathode. If we now assume that the potential profile for an *expanding* thruster beam is obtained by a self-similar expansion of the electron diode-like

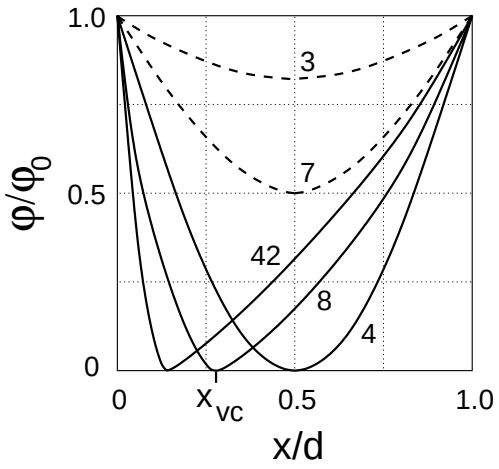

Figure 4.15: Schematic representation of potential trough solutions (dashed lines) and virtual cathode solutions (solid lines) for the potential profile in the interelectrode space of a short-circuited electron diode for various values of α (modified after Birdsall and Bridges [1966]).

solution according to the velocity of the ion beam, we can qualitatively explain the finite velocity of the shock front as well as its dependence on the beam width: when the beam expands self-similarly, with the front end moving at $v = v_{i0}$, the virtual cathode or shock front that is located at x_{vc} in the stationary case, adopts a velocity $v_{sh} = v_{i0}x_{vc}/d$. As $x_{vc} < d$, the shock front is always slower than the front end of the ion beam.

Moreover, Birdsall and Bridges [1966] find that x_{vc}/d decreases for increasing α (cf. Fig. 4.15). Transferred to the ion thruster case, this would mean a decrease in shock velocity for increasing α. According to Eqn. (4.12), α is proportional to the electron current density J_0, which can be calculated by taking the downstream values for electron density and electron velocity:

$$\alpha \propto J_0 = en(x)v(x) = en_{down}v_{down} = en_{down}v_{i0} \ . \tag{4.14}$$

It follows that $\alpha \propto n_{down}$, which means that higher downstream electron densities should result in lower shock velocities. This is exactly what we observe in our simulations (lower panel of Fig. 4.12).

4.1.4 Shock-like vs. shock-free neutralization

The above investigations revealed that the neutralization regime switches between two different scenarios, depending on the injection velocity ratio $\eta = v_{e0}^{th}/v_{i0}$. For $\eta < 1.7$, a moving electrostatic shock of a few kT_{e0}/e provides thermalization, while there is no such shock for $\eta > 1.7$. A necessary condition for the formation of such a shock was derived (Eqn. 4.6) on the basis of a simple 1D model (Fig. 4.6) and was found to be consistent with the simulation results.

In both scenarios, the potential profile exhibits a sheath adjacent to the injection plane, where the electrons are accelerated to form a beam at $2 - 3v_{i0}$. The beam strength increases as η is reduced. The passage of this beam through the hot electron component that is trapped between the injection plane and the shock / the front end of the ion beam gives rise to beam Langmuir oscillations. Further downstream, the electron plasma drifts with the ion bulk velocity, independent of η. However, in contrast to the case $\eta < 1.7$, the downstream electron velocity distribution for the shock-free case is not Maxwellian, but rather exhibits a double-peaked shape, corresponding to the weakened left-over of the electron beam and its reflection at the front end of the ion beam.

For the case $\eta = 1$ we also studied the behaviour of the shock front for varying lateral beam dimensions b. An increasing beam width resulted in increasing degrees of neutralization n_e/n_{i0} and in a decreasing velocity of the shock front. Upon approaching complete neutralization, i. e. $n_e/n_{i0} = 1$, the shock front seems to become stationary. We presented a tentative model, which regards an ion thruster beam as a self-similarly expanding electron diode and identifies the shock front as a virtual cathode. This model can qualitatively explain the observed variation of the shock velocity.

The technical importance of the shock wave that builds up when condition (4.6) is met lies in the fact that it generates a fully thermalized plasma downstream with practically no residual axial electric field and hence no free energy to drive unwanted plasma-instabilities. Ion thrusters undergoing such a shock-neutralization would therefore induce less electromagnetic interference – one of the major technical concerns of the operation of electric propulsion, the other being the erosion of the thruster grid by backstreaming CEX ions (Ch. 1). Also in this respect, the shock-like neutralization scenario is superior to the shock-free one: An inherent feature of the shock is the potential drop of a few $k_B T_{e0}$. This drop acts as a "shield" against backstreaming CEX ions, whose energies are typically below $0.05 k_B T_{e0}$ [Wang et al., 1996], and can thus significantly reduce the erosion damages on the thruster.

The occurrence of the shock wave requires $\eta < 1.7$, whereas current ion thrusters are operating at $\eta \gg 1$. Given the recent interest in ion engines, future thrusters might, however, be capable of accessing the range around $\eta = 1.7$. Such thrusters would fulfill the necessary condition for a shock-like neutralization and could then benefit from a very efficient neutralization.

For these reasons, shock-like neutralization appears to be a promising option for the design of new ion thrusters. What has to be checked, however, is if the shock wave still builds up and how it behaves for more realistic conditions than those simulated so far. Among the aspects that have to be addressed in this respect are the influence of a static magnetic field, which is commonly encountered in ion thruster environments (see Sec. 4.2), and the spatial separation between the ion source and the electron emitting cathode (Sec. 4.3).

4.2 The impact of an axial magnetic field

This section is dedicated to investigating the effect of a static magnetic field on the shock-like neutralization described in the foregoing. The solar wind with roughly 5 nT at 1 AU can be neglected as a source of a static magnetic field. Much stronger fields arise from permanent magnets in the DS1 thruster chamber. An upper limit for the field generated by the DS1 permanent magnets is 5×10^{-4} T at the thruster exit [Richter, 2000; Brinza et al., 2000]. We do, however, not restrict ourselves to an assessment of the impact of a magnetic field that is comparable in strength to the actual DS1 field, but also explore the possibility of generating favorable conditions for the technically interesting shock-like neutralization by applying much stronger magnetic fields.

The simulation setting is identical to the one of the previous section. In particular, we still disregard the spatial separation between electron and ion source, and focus on the electron time scale by using the actual electron to ion mass ratio. However, in contrast to the simulations of Sec. 4.1, we now apply a static magnetic field in the $+x$ direction. Its strength is measured in terms of the ratio ψ between electron gyrofrequency and electron plasma frequency $\psi := \Omega_e/\omega_{pe}^0$. The actual field strength for the DS1 engine of about 5×10^{-4} T at the thruster exit corresponds to $\psi = 0.02$. In our simulations, however, we cover a ψ-range of $0.1 - 1.0$, i. e. we simulate much stronger magnetic fields, because it is this range, where the effects of the magnetic field become significant.

Apart from that, some runs are produced by applying $\psi = 10^9$, which yields gyroperiods of 2×10^{-7} and 5×10^{-2} time steps for electrons and ions, respectively. As demonstrated in Sec. 2.4, magnetic fields of this strength result in a "shaking" of the particles, with their perpendicular velocity components being reversed after each time step. It reproduces the actual effect of a very strong magnetic field in that it inhibits a net motion across $\mathbf{B}$. The particles, however, do not perform a circular gyromotion but a linear motion. Rather than the analytically determined gyroradius, it is the spatial extent $v_\perp \Delta t$ of this linear motion that is to be taken as a measure of the effective "gyroradius" that is actually being simulated. In the case of a magnetic field strength of $\psi = 10^9$, it amounts to negligible $\Delta X/25$ for electrons with a perpendicular velocity of $v_\perp = v_{e0}^{th}$ and to $\Delta X/1250$ for ions with $m_i = 250,000 m_e$ and $T_i = T_e$. Therefore, these runs serve as a proxy to the case of infinitely strong magnetic fields, and the field strength corresponding to $\psi = 10^9$ is from now on referred to as "infinite".

4.2.1 Infinite axial magnetic field

As a first step to explore the general effects of an axial magnetic field on ion thruster beam neutralization, we apply an "infinitely" strong field for the runs of this section. The results for $\eta = 1$ and $b = 10\lambda_D$ are presented in Figs. 4.16-

4.18. For the diagnostics, we adopt the same averaging and normalization as in the previous section: The electron density is averaged across the beam and normalized to the nominal ion density of a non-diverging beam n_{i0}, while the potential is obtained by averaging the axial component of the electric field E_x across the beam, integrating in axial direction and setting $\Phi = 0$ in the injection plane ($x = 10$). It is normalized to $\Phi_0 := k_B T_{e0}/e$.

For comparison, we briefly recall the main results for the corresponding simulation run in the absence of a magnetic field (Sec. 4.1.1). For $B = 0$, the plasma beam broke down into five regions (Fig. 4.3):

1. The acceleration region of about $10\lambda_D$ thickness, where the potential rises steeply and the injected electrons are accelerated to about twice the ion bulk velocity.
2. The wavy high potential beam region with $\Phi \approx 3\Phi_0$. The electrons in this region are roughly equipartitioned between a strong beam at $v \approx 2v_{i0}$ and a trapped hot component.
3. The thermalization region: A shock front of 10 to $15\lambda_D$ thickness, where the potential drops by $2.3\Phi_0$, and the electron density jumps by a few percent of n_{i0}.
4. The quiescent region of constant potential at $\Phi = 0.75\Phi_0$ with a Maxwellian electron component drifting at v_{i0}.
5. The vacuum region ahead of the ion beam with a strongly decreasing electron density.

As shown in Fig. 4.16, the infinite magnetic field leads to a different structure of the beam plasma:

1. An injection sheath, where the electrons are accelerated to about $2v_{i0}$ builds up similar to the $B = 0$ case.
2. There is no extended high potential region corresponding to the beam region of the $B = 0$ case.
3. Instead of a single shock front with complete thermalization, a series of potential hills of a few $k_B T_{e0}/e$ forms, with a spatial repetition rate of about $12\lambda_D$, i. e. of roughly one electron inertia length l_e. The hill amplitudes decrease in the downstream direction, until they finally vanish around $x = 90$. Within each of the potential hills, the electron plasma consists of a beam component and a trapped component, similar to the high potential region for $B = 0$. As the strong axial magnetic field inhibits a net particle motion in the perpendicular direction and thus enforces perfect mass continuity in the x direction, the electron density strongly anticorrelates with potential and electron beam velocity. The density perturbations are highly non-linear and reach a maximum peak-to-peak amplitude of about $0.4n_{i0}$.

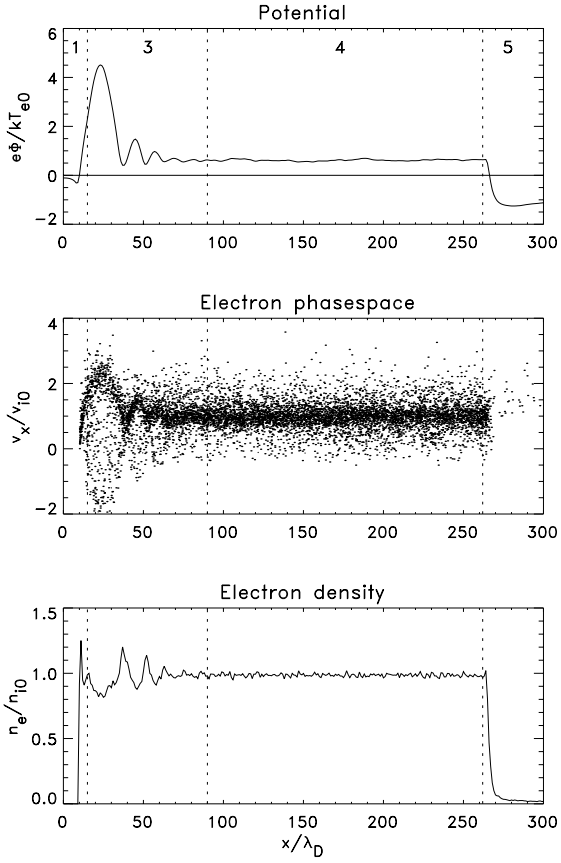

Figure 4.16: Potential, electron phase space, and electron density for $\eta = 1$, $b = 10\lambda_D$, $\omega_{pe}^0 t = 200$, and $B \to \infty$. The different regions are numbered analogously to Fig. 4.3.

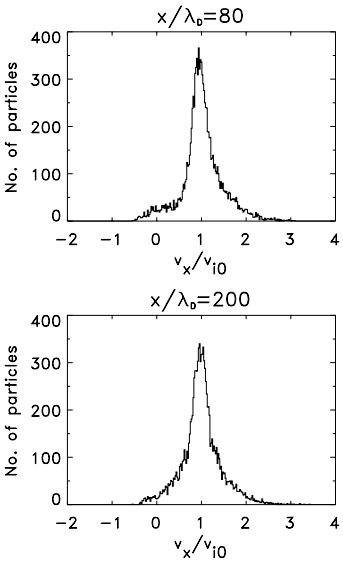

Figure 4.17: Electron velocity distribution function around the disappearance of the potential hills ($x = 80\lambda_D$, upper panel) and further downstream ($x = 200\lambda_D$, lower panel) for $\eta = 1$, $b = 10\lambda_D$, $\omega_{pe}^0 t = 200$, and $B \to \infty$.

4. A few l_e behind the decay of the potential hills, the plasma becomes fully thermalized: a drifting Maxwellian with a drift velocity equal to v_{i0} (cf. Fig. 4.17). The potential reaches a stationary level of $0.6\Phi_0$, a value similar to the $B = 0$ case. Noting that the majority of electrons are injected with $v_e = 0$, this value of Φ corresponds roughly to the energy that is necessary to accelerate them to v_{i0}. Therefore, it does not surprise that this potential value is rather independent of whether an axial magnetic field is applied or not. The electron density, however, differs considerably from the $B = 0$ case: Since the infinite magnetic field effectively inhibits the escape of electrons in the lateral directions, almost perfect space charge neutralization is achieved on the downstream side with an electron density of about $0.98n_{i0}$.

5. In the region ahead of the ion beam, the potential falls down to an ambient level of $-1.3\Phi_0$, as compared to $-3.8\Phi_0$ for the case $B = 0$. This more moderate value is due to the much higher degree of neutralization.

The temporal development of these regions for infinite magnetic field is presented in Fig. 4.18, which shows contours of the axial electric field E_x. As can be seen in this Figure, the potential hills are time stable and are almost stationary. They are moving with a uniform velocity $v_p = 0.07v_{i0}$, which is much slower than the single shock wave of the $B = 0$ case ($v_{sh} = 0.38v_{i0}$ for $b = 10\lambda_D$).

The white line in Fig. 4.18 marks the approximate boundary between non- and fully thermalized plasma, as obtained from electron phase space analyses. While for $B = 0$ there was a sharp transition between these two plasma states, the electron velocity distribution changes in a more continuous manner when an infinite magnetic field is applied. Apparently, it requires several potential hills in order to thermalize the electrons. Therefore, the beginning of the thermalized section of the beam cannot be determined precisely, and the white line in Fig. 4.18 is to be understood as a rough estimate only. What can be seen, however, is that the region of thermalized plasma is an opening cone, much as it is for the case of no magnetic field (cf. Sec. 4.1.1).

Concerning the source of the series of potential hills, we refer to the considerations of Sec. 4.1.1 on the excitation of the single shock wave and its associated potential jump in the $B = 0$ case. Based upon the similarity of that shock to an electrostatic double layer, it was speculated that the same mechanism that gives rise to double layers might also be responsible for the formation of the observed shock wave: the reflection of electrons at a negative charge density spike [Hasegawa and Sato, 1982] or, in the case of the ion thruster, at the leading edge of the expanding ion beam. Fig. 4.18 suggests that this explanation is also applicable to the potential hills here, which apparently originate from the leading edge of the ion beam. Considering, however, that the infinite magnetic field confines the plasma movement to one direction, these excursions in E_x can be better understood as a damped, non-linear electrostatic wave in a *one-dimensional* plasma, which is excited by the difference between electron and ion bulk velocity upon injection. This will be shown in the following.

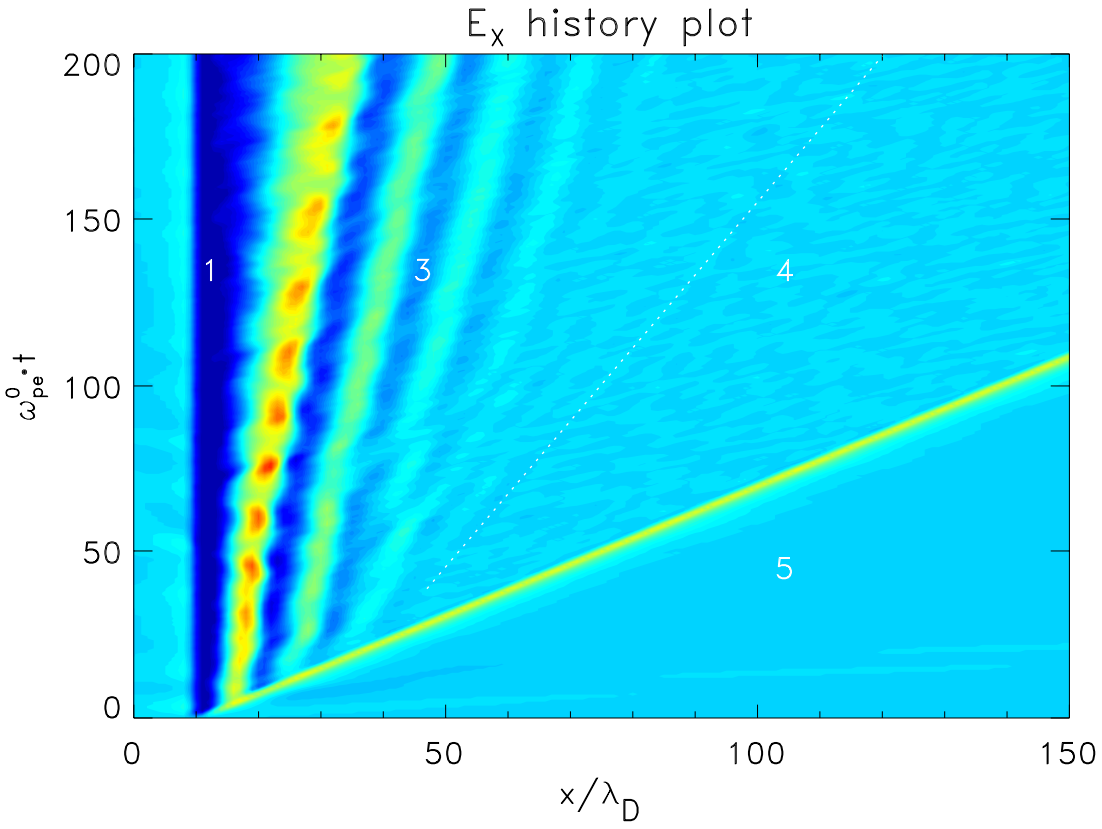

Figure 4.18: History plot of the axial electric field E_x for $\eta = 1$, $b = 10\lambda_D$, and $B \to \infty$. The different regions are numbered as in Fig. 4.16. Note the acceleration sheath (1), the series of excursions in E_x (3), the region of thermalized plasma (4) and the vacuum in front of the ion beam (5). The blue tone in region (5) corresponds to zero electric field, and "hotter" and "colder" colours to positive and negative E_x, respectively. The dotted white line indicates the beginning of the fully thermalized plasma as obtained by electron phase space analyses.

We start from the one-dimensional divergence equation for the electric field E, the electron equation of motion, and the continuity equation:

$$\frac{dE}{dx} = \frac{e}{\varepsilon_0}(n_i - n_e) \tag{4.15}$$

$$m_e v_e \frac{dv_e}{dx} = -eE \tag{4.16}$$

$$n_e v_e = n_{i0} v_{i0}. \tag{4.17}$$

Here E, v_e, and v_i are the components of the respective vectors in x direction. In Eqn. (4.17), we have assumed that electrons and ions are injected with the same number per time step, and that – by virtue of the infinite magnetic field – no electrons are lost through the lateral beam boundaries.

We differentiate Eqn. (4.16) and substitute dE/dx from Eqn. (4.15) to obtain

$$v_e \frac{d^2 v_e}{dx^2} + \left(\frac{dv_e}{dx}\right)^2 = -\frac{e^2 n_i}{m_e \varepsilon_0}\left(1 - \frac{n_e}{n_i}\right). \tag{4.18}$$

Noting that $n_i \equiv n_{i0}$ for the time interval covered by our simulations, and that

$$\frac{e^2 n_{i0}}{m_e \varepsilon_0} = \left(\omega_{pe}^0\right)^2, \tag{4.19}$$

we can substitute n_e/n_i with Eqn. (4.17) to end up with

$$v_e^2 \frac{d^2 v_e}{dx^2} + v_e \left(\frac{dv_e}{dx}\right)^2 + \left(\omega_{pe}^0\right)^2 (v_e - v_{i0}) = 0. \tag{4.20}$$

We normalize the electron velocity to the ion bulk velocity by defining $u := v_e/v_{i0}$, and introduce a spatial scale $\xi := v_{i0}/\omega_{pe}^0$ for the derivatives:

$$(\cdot)' := \frac{d}{d(x/\xi)} = \xi \frac{d}{dx}. \tag{4.21}$$

With these definitions, Eqn. (4.20) reads

$$u^2 u'' + u u'^2 + u - 1 = 0. \tag{4.22}$$

This non-linear equation describes the electron velocity for static electron density perturbations in a neutralizing ion background that is streaming with velocity 1.

A trivial solution of Eqn. (4.22) is $u \equiv 1$, which corresponds to equal electron and ion bulk velocities, i. e. to a quasi-neutral plasma moving with the bulk velocity 1. In order to obtain an impression of the non-trivial solutions, we first linearize Eqn. (4.22) by writing u as $u_0 + u_1$ with $u_1 \ll u_0$ and setting u_0 equal to the "equilibrium" value $u_0 = 1$. Substituting this into Eqn. (4.22) and neglecting terms of second order in u_1 leads to

$$u_1'' + u_1 = 0. \tag{4.23}$$

Hence, the solutions for u in the linear case are

$$u = (1 + \hat{u}) \sin x, \tag{4.24}$$

with a certain amplitude $\hat{u}$. When rescaled, these solutions correspond to sinusoidal perturbations of v_e around the equilibrium velocity v_{i0} with a periodicity of $2\pi\xi = 2\pi v_{i0}/\omega_{pe}^0$.

In order to obtain exact solutions of the non-linear Eqn. (4.22), we integrated this equation numerically with a fourth-order Runge-Kutta method. As boundary conditions at $x = 0$ we used $u' = 0$ and varied the initial electron to ion velocity ratio $u = v_e/v_{i0}$ between 1 and 1.5. For a better comparison with the simulation

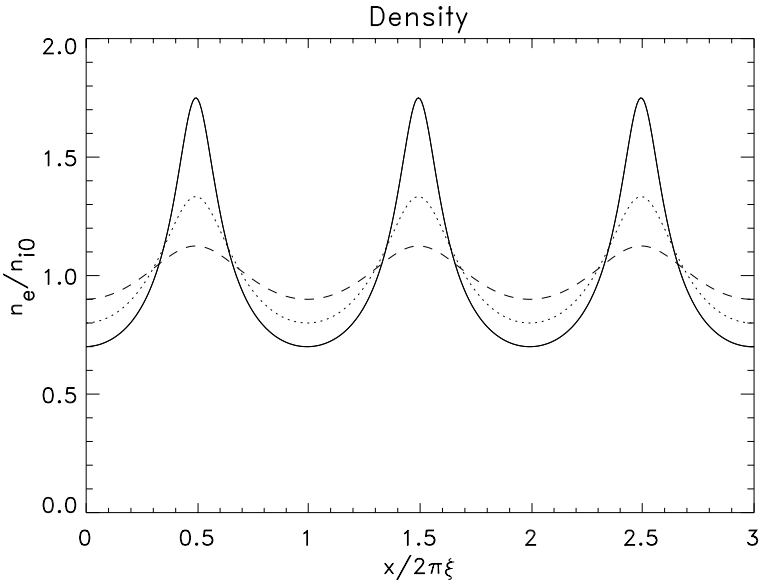

Figure 4.19: Density profiles for $n_e/n_{i0}|_{x=0} = 0.7$ (dashed), 0.8 (dotted), and 0.9 (solid), as obtained by integrating Eqn. (4.22) with a fourth-order Runge-Kutta method. Note the $2\pi\xi$-periodicity and the characteristic shape with sharp maxima and "round" minima.

results, we employed Eqn. (4.17) to convert the obtained velocity solutions into density profiles. As can be seen from Fig. 4.19, which shows the density profiles for $n_e/n_{i0}|_{x=0} = 0.7$, 0.8, and 0.9, the exact solutions exhibit a periodicity of $2\pi\xi$, similar to the solutions of the linearized equation.

For the simulation runs with $\eta = v_{e0}^{th}/v_{i0} = 1$, the spatial scale of $2\pi\xi$ would correspond to

$$2\pi\xi = 2\pi v_{i0}/\omega_{pe}^0 = 2\pi v_{e0}^{th}/\omega_{pe}^0 = 2\pi\sqrt{2}\lambda_D \approx 9\lambda_D. \tag{4.25}$$

This is of the same order as the observed periodicity of about $12\lambda_D$ of the nonlinear density perturbations in Fig. 4.16 (and later in Fig. 4.23). Moreover, the shape of the potential profiles as obtained by simulation and numerical integration, respectively, with their pointed maxima and rather "round" minima are quite similar.

While the basic equations (4.15)–(4.17) used to derive Eqn. (4.22) assume a cold plasma, the simulation involves a finite thermal velocity of both electrons and ions. The finite electron temperature gives rise to Landau damping and therefore, the amplitude of the density perturbations decreases in the downstream direction, in contrast to the constant amplitude of the cold plasma solutions of Eqn. (4.22).

The decrement γ/ω_{pe}^0 for Landau damping is

$$\gamma/\omega_{pe}^0 = \frac{\pi}{2} \cdot \left(\frac{\omega_{pe}^0}{k}\right)^2 \cdot \left[\frac{\partial f}{\partial v}\right]_{v=v_\varphi} , \tag{4.26}$$

where k is the wave number, v_φ the phase velocity of the wave and f the distribution function of the electrons [Chen, 1974, p. 245]. If we assume the electrons to have a Maxwellian distribution, which is not exactly the case, the derivative can be computed as

$$\left[\frac{\partial f}{\partial v}\right]_{v=v_\varphi} = -\frac{2v_\varphi}{\sqrt{\pi} v_{th}^3} \cdot \exp\left(-\frac{v_\varphi^2}{v_{th}^2}\right) , \tag{4.27}$$

with v_{th} being the thermal velocity of the electrons within the wave. The observed periodic structure is quasi-stationary; its phase velocity v_φ in the frame of reference of the ion beam is therefore v_{i0}. Noting further that

$$\omega_{pe}^0 = \frac{1}{\sqrt{2}} \frac{v_{e0}^{th}}{\lambda_D} \tag{4.28}$$

and that $v_{e0}^{th} = v_{i0}$ for the simulated case of $\eta = 1$, the decrement can be written as

$$\gamma/\omega_{pe}^0 = -\frac{\sqrt{\pi}}{2} \cdot \frac{1}{k^2\lambda_D^2} \cdot \left(\frac{v_{i0}}{v_{th}}\right)^3 \cdot \exp\left(-\frac{v_{i0}^2}{v_{th}^2}\right) . \tag{4.29}$$

Substituting $k \approx 2\pi/(12\lambda_D)$ (see above) and $v_{th} \approx 0.5 v_{i0}$, Eqn. (4.29) yields a decrement of

$$\gamma/\omega_{pe}^0 \approx -0.5 , \tag{4.30}$$

meaning that after about 2 oscillations, the amplitude should decrease by a factor of e. Judging from Fig. 4.16, or better from Fig. 4.23, this is of the right order.

Hence, due to the infinite magnetic field the plasma behaves essentially one-dimensional, and the series of potential hills can be understood as a non-linear, Landau-damped electrostatic wave excited by the difference between electron and ion bulk velocity upon injection.

Dependence on velocity ratio

In the absence of a magnetic field, a strong dependence of the neutralization scenario on the injection velocity ratio η was found (Sec. 4.1.1). Based upon the simple 1D potential step model of Fig. 4.6, we were able to derive Eqn. (4.6) as a necessary condition for the neutralization to be shock-like:

$$v_{i0} > v_e^{th} . \tag{4.6}$$

That is, the ion beam velocity has to be greater than the electron thermal velocity on the downstream side in order for a shock structure to develop. In a series of

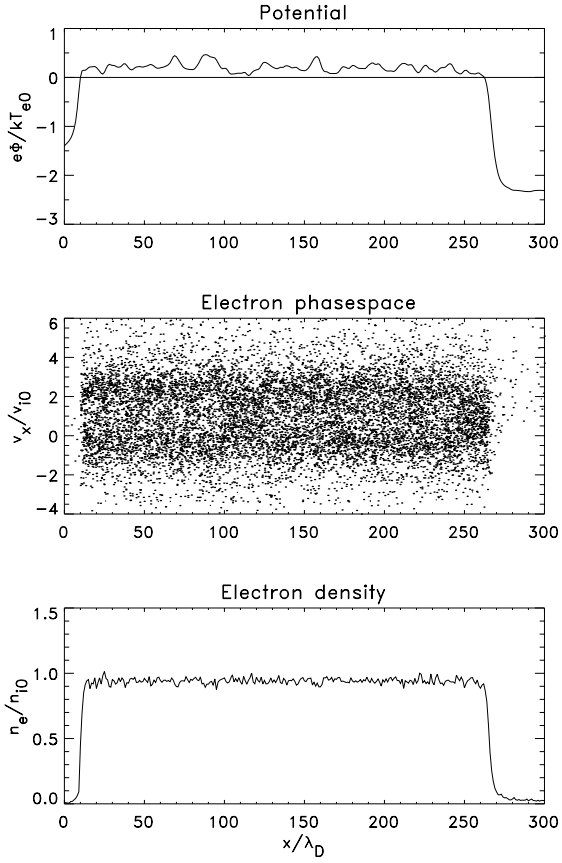

Figure 4.20: Potential, electron phase space, and electron density for $\eta = 3$, $b = 10\lambda_D$, $\omega_{pe}^0 t = 540$, and $B \to \infty$.

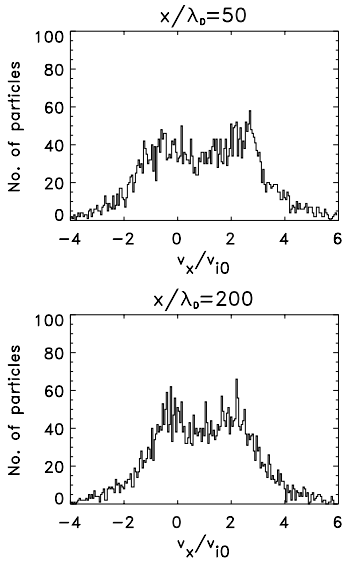

Figure 4.21: Electron velocity distribution function close to the injection plane ($x = 50\lambda_D$, upper panel) and further downstream ($x = 200\lambda_D$, lower panel) for $\eta = 3$, $b = 10\lambda_D$, $\omega_{pe}^0 t = 540$ and $B \to \infty$.

simulation runs with different injection velocity ratios $\eta = v_{e0}^{th}/v_{i0}$, we were able to confirm the validity of condition (4.6), and showed that for $\eta > \eta_{crit} \approx 1.7$ shock-free solutions occur.

Here, in the case of an infinite magnetic field, no single shock wave but a series of potential hills forming a damped electrostatic wave develop. Our simple 1D model, however, is applicable to each of these potential hills, so that the necessary condition (4.6) should be valid for them as well. The relevant thermal velocity to enter Eqn. (4.6) is the axial component (x) of the electron thermal velocity, which amounts to $v_x^{th} = 0.46 v_{i0}$ for the simulation run of Fig. 4.16. This results in a critical injection velocity ratio of $\eta_{crit} = v_{e0}^{th}/v_{i0} = 1/0.46 = 2.2$. For $\eta < \eta_{crit}$, a potential profile as in Fig. 4.16 should show up, while simulations with $\eta > \eta_{crit}$ should produce shock-free solutions.

We verified that by first simulating the case $\eta = 2$. In accordance with the above derived η_{crit}, the solution looks still like that in Fig. 4.16, with a series of potential hills and a Maxwellian plasma downstream. The potential excursions and the associated density enhancements were, however, much weaker than for $\eta = 1$. This marks already the transition to the shock-free solution that is obtained when η is further increased: Figs. 4.20 and 4.21 show the results for the case $\eta = 3$. In perfect consistency with Eqn. (4.6), there is indeed no sign of a shock-like structure. As compared to the potential excursions of up to $4.5\Phi_0$ for $\eta = 1$, the potential here shows only moderate fluctuations around $\Phi = 0.2\Phi_0$. And in contrast to the strongly non-linear density perturbations in the $\eta = 1$ case, the density for $\eta = 3$ stays rather constant at $n_e = 0.93 n_{i0}$.

Also the phase space looks considerably different (cf. Fig. 4.21): Close to the injection plane, the electron velocity distribution is a double-peaked function with one electron beam at $-0.8 v_{i0}$ and the other at $2.6 v_{i0}$. While the latter arises through acceleration in the injection sheath, similar to the beam for $\eta = 1$, the former is probably due to continuous reflection of these beam electrons at the front end of the ion beam. Further downstream, the two peaks tend to merge. However, the electron distribution function does not become Maxwellian.

Despite the lack of a damped electrostatic wave providing thermalization, the electrons adapt their overall drift velocity to the ion beam velocity v_{i0}, similar to what was observed in the shock-free solution for $B = 0$. As noted already, in the absence of a shock, this adaptation can take place via two-stream instability between the two electron beams appearing in Fig. 4.21 or via a quasi-Fermi-deceleration, which will be discussed in detail in Sec. 4.3.2.

Hence, the "switching" between two different neutralization scenarios depending on η, which was already observed in the absence of a magnetic field, takes place in a similar manner when an axial magnetic field is applied. Also Eqn. (4.6) that allows to determine the critical injection velocity ratio η_{crit} seems to be applicable to the $B \neq 0$ case; however, due to the different downstream electron temperature, with a different η_{crit} than for $B = 0$. The apparent dependence of the downstream temperature on the magnetic field will be explored in more detail in Sec. 4.2.2.

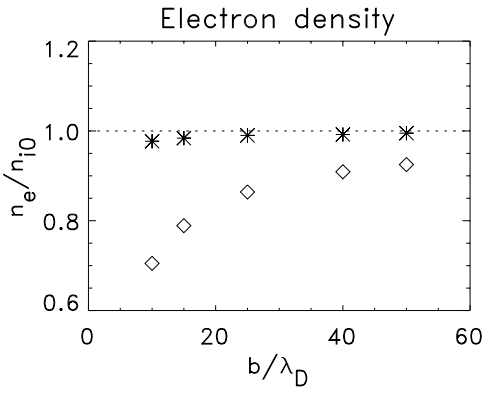

Figure 4.22: Dependence of the downstream electron density on the beam width for zero magnetic field ($\diamond$, cf. Sec. 4.1.2) and for infinite magnetic field ($*$). The dotted line represents complete neutralization.

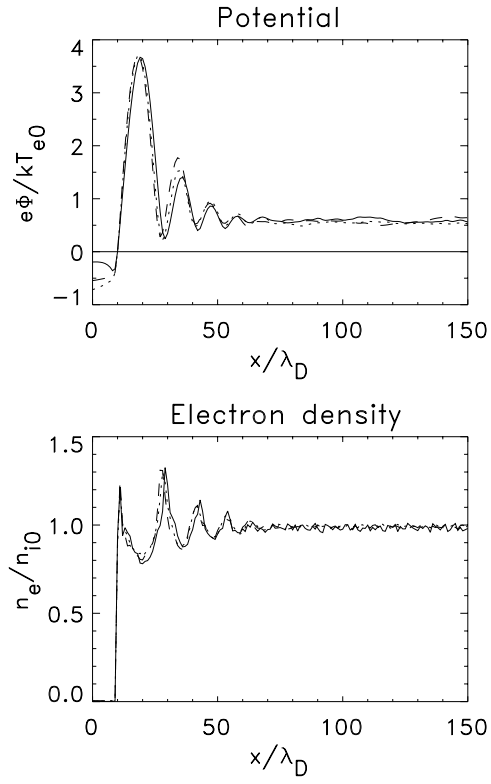

Figure 4.23: $\eta = 1$, $\omega_{pe}^0 t = 120$, and $B \to \infty$: Potential and electron density for $b = 10\lambda_D$ (solid line), $25\lambda_D$ (dashed), and $40\lambda_D$ (dotted).

Influence of lateral beam dimension

Without an axial magnetic field, the finiteness of the lateral beam dimension was found to have a significant impact on the overall beam behaviour (Sec. 4.1.2). While the general morphology of the solution turned out to be quite independent of the beam width b, we found a strong dependence on b of the downstream electron density (cf. Fig. 4.22) and of the shock front velocity, ranging from $v_{sh} = 0.5v_{i0}$ for $b = 5\lambda_D$ to $0.19v_{i0}$ for $b = 50\lambda_D$ (Fig. 4.12).

In the case of an infinite magnetic field, however, a variation of the beam width has only minor effects. As can be seen from Fig. 4.23, which shows the potential profile and the electron density for beam widths of $b = 10, 25$ and $40\lambda_D$ for $\psi = 10^9$, the respective curves are very similar. The downstream potential level is around $0.6\Phi_0$, and, due to the B-field enforced immobility of the electrons in the lateral directions, the respective downstream densities get close to n_{i0} in all cases (cf. Fig. 4.22). There is only a slight variation in the amplitude of the excursions of potential and density. Their velocity is of the order of $0.07v_{i0}$ for all investigated beam widths b, which is much smaller than the shock velocity of the $B = 0$ case.

At first glance, the strong dependence of the shock velocity on the beam width in the absence of a magnetic field contrasts with the rather weak impact of b in the case of an infinitely strong magnetic field. However, rather than being directly due to the finiteness of the beam in the lateral direction, we could attribute the variation of the shock velocity for $B = 0$ to the different degrees of neutralization that came along with varying beam widths. Hence, considering that an infinite axial magnetic field inhibits electron escape through the lateral beam boundaries and thus enforces almost perfect space charge neutralization independent of the beam width, it does actually not surprise that a variation of the beam width has practically no effect on the overall beam behaviour in the case of an infinitely strong magnetic field.

4.2.2 Finite axial magnetic fields

After having investigated the impact of an axial magnetic field of infinite strength on the neutralization process, we now focus on the effects of intermediate field strengths. In order to illustrate the transition between the two extreme cases of $\psi = 0$ and $\psi \to \infty$, the potential, the electron phase space, and the electron density for a simulation run corresponding to the one of Fig. 4.16, but with $\psi = 1$ instead of $\psi = 10^9$, are shown in Fig. 4.24. Apparently, the plasma beam exhibits elements of both extreme cases.

As compared to the case of infinite B, only a single potential excursion builds up, while the subsequent potential hills of Fig. 4.16 have disappeared. The downstream potential level, however, is still $\Phi = 0.6\Phi_0$.

In contrast to the series of strongly non-linear perturbations of the density for the case of infinite B, the density here rather exhibits a step from $0.65n_{i0}$ within

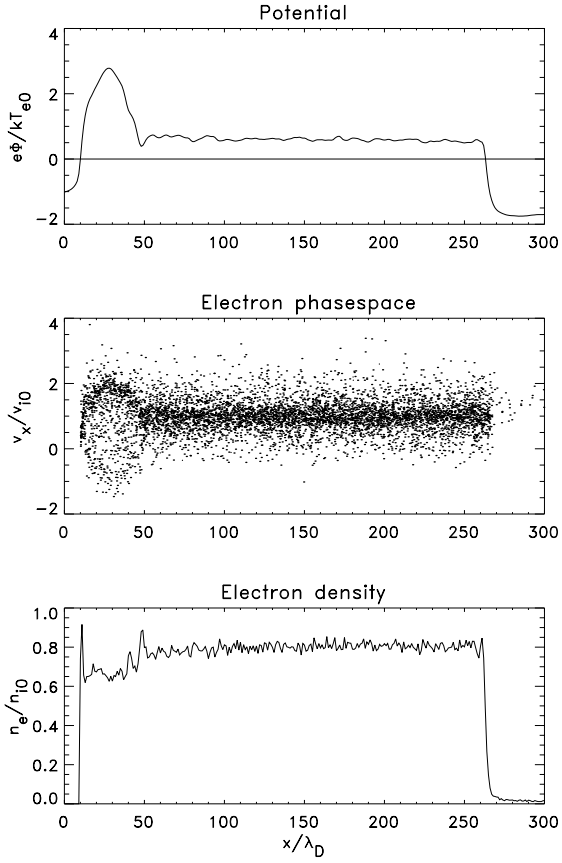

Figure 4.24: Potential, electron phase space, and electron density for $\eta = 1$, $b = 10\lambda_D$, $\omega_{pe}^0 t = 200$, and $\psi = 1$.

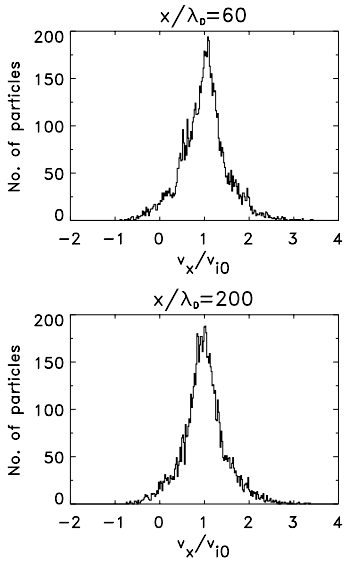

Figure 4.25: Electron velocity distribution function right behind the potential hill ($x = 60\lambda_D$, upper panel) and further downstream ($x = 200\lambda_D$, lower panel) for $\eta = 1$, $b = 10\lambda_D$, $\omega_{pe}^0 t = 200$, and $\psi = 1$.

the potential hill to $0.81n_{i0}$ downstream. In this respect, it behaves similar to the $B = 0$ case (cf. Fig. 4.3), where the density jumps by a few percent at the location of the shock front. As can be expected from the confining character of the axial magnetic field, the downstream density of $n_e = 0.81n_{i0}$ for $\psi = 1$ lies between the values for $\psi = 0$ and $\psi = 10^9$ ($0.7n_{i0}$ and $0.98n_{i0}$, respectively).

The electrons are not fully thermalized directly behind the potential drop, as they are in the absence of a magnetic field, but rather require several electron inertia lengths before ultimately adopting a Maxwellian distribution (Fig. 4.25), similar to the case of infinite B in Sec. 4.2.1.

As seen in the previous section, the confinement of the electron movement to the axial direction in the case of infinite magnetic field causes the plasma beam to behave essentially one-dimensional. For finite magnetic fields, however, this one-dimensionality breaks down, because the electrons acquire a certain mobility in the direction perpendicular to $\mathbf{B}$. They can leave the beam through the lateral beam boundaries, resulting in a loss term in the continuity equation (4.17) and can redistribute themselves over the beam cross section, thereby changing the amount of electrical flux through the lateral surfaces (Eqn. 4.15). Hence, for finite magnetic fields, the electrons have more degrees of freedom to adjust themselves to the non-equilibrium injection conditions. In contrast to the $\psi \to \infty$ case, they are not forced to build up a 1D series of potential hills, which is the only way of responding to the initial difference between electron and ion injection velocity in a 1D plasma. This explains the disappearance of the non-linear electrostatic wave upon reduction of the magnetic field strength from $\psi \to \infty$ to $\psi = 1$.

What the potential looks like when the magnetic field strength is further reduced, is displayed in Fig. 4.26, which shows the potential profiles for $\psi = 0, 0.2, 0.5, 1,$ and 10^9 for a beam width of $b = 10\lambda_D$ and an injection velocity ratio of $\eta = 1$. It illustrates the transition between the series of potential excursions for infinite B to the extended high potential region with a single shock for $B = 0$. While the *way* in which the electron plasma gets thermalized (i. e. either via a single shock wave, a series of potential excursions or a mixture of both) apparently depends on the magnetic field strength, the potential level in the region of thermalized plasma is quite independent of B and covers only a small range between $0.6\Phi_0$ and $0.75\Phi_0$. As stated above, this value is mainly determined by the energy that is necessary to accelerate the injected electrons, most of which are born with $v_e = 0$, to the ion bulk velocity v_{i0}. Since this is the same for all B, the downstream potential level should indeed not vary too much.

As can also be seen from Fig. 4.26, the potential profile for $\psi = 0.2$ looks already very similar to the one for $\psi = 0$. Hence, for a significant impact of the magnetic field on the neutralization scenario, a minimum field strength around $\Omega_e = 0.2\omega_{pe}^0$ is required. A magnetic field corresponding to $\psi = 0.2$ would amount to about 5×10^{-3} T in the case of DS1, which is 10 times stronger than the actual field. It can be deduced that typical spacecraft fields have practically no effect on the shock-like neutralization.

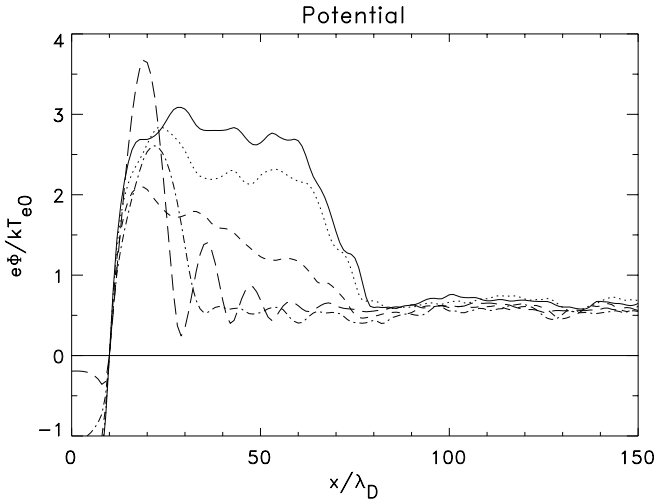

Figure 4.26: $\eta = 1$, $b = 10\lambda_D$, and $\omega_{pe}^0 t = 120$: Potential profiles for $\psi = 0$ (solid line), 0.2 (dotted), 0.5 (dashed), 1 (dashed-dotted), and 10^9 (long dashes).

Dependence of downstream state on field strength

As demonstrated above, common spacecraft stray fields are unlikely to influence the neutralization process significantly. The question arises, however, if the *intentional* application of much stronger magnetic fields can generate favorable conditions for an optimal neutralization. Therefore, the dependence of the downstream plasma state on the applied magnetic field for field strengths beyond typical spacecraft values deserves further attention.

An "optimal" neutralization would have the following characteristics: (a) a fully thermalized plasma on the downstream side, (b) a low degree of non-neutrality, and (c) small residual electric fields, i. e. no major potential fluctuations.

The requirement of a thermalized plasma with no free energy to drive unwanted plasma instabilities favours the shock-like neutralization. For this neutralization scenario, the amplitudes of the potential fluctuations on the downstream side are quite small and turned out to be independent of both the beam width b (Sec. 4.1.2) and the magnetic field strength (Fig. 4.26). However, in order to operate the thruster in the shock regime, the necessary condition Eqn. (4.6) has to be fulfilled. As the downstream electron temperature enters this relation, it is worthwhile to have a look to its dependence on the magnetic field strength.

The downstream parallel and perpendicular electron temperatures of a series of simulation runs with varying beam widths and different magnetic field strenghts are shown in Fig. 4.27. They are normalized to the injection temperature T_{e0} and are displayed as functions of the normalized magnetic field strength ψ (left column) and of the ratio between electron gyroradius and beam width r_{ge}/b (right column). The injection velocity ratio is $\eta = 1$ for all runs, and the gyroradius used here is the thermal gyroradius, i. e. $r_{ge}(v_\perp = v_{e0}^{th})$.

Fig. 4.27 comprises the dependence of the respective temperatures both on magnetic field strength B and on beam width b. While the parallel temperature $T_{e\|}$ is rather independent of b and shows a consistent behaviour when plotted versus the magnetic field strength, $T_{e\perp}$ can be better described as a function of r_{ge}/b, which involves both B and b. For a given beam width b, the two temperatures respond oppositely to an increasing magnetic field: $T_{e\|}$ drops from $0.32T_{e0}$ for $B = 0$ to $0.21T_{e0}$ for infinite B, while $T_{e\perp}$ rises significantly from around $0.43T_{e0}$ to $0.95T_{e0}$.

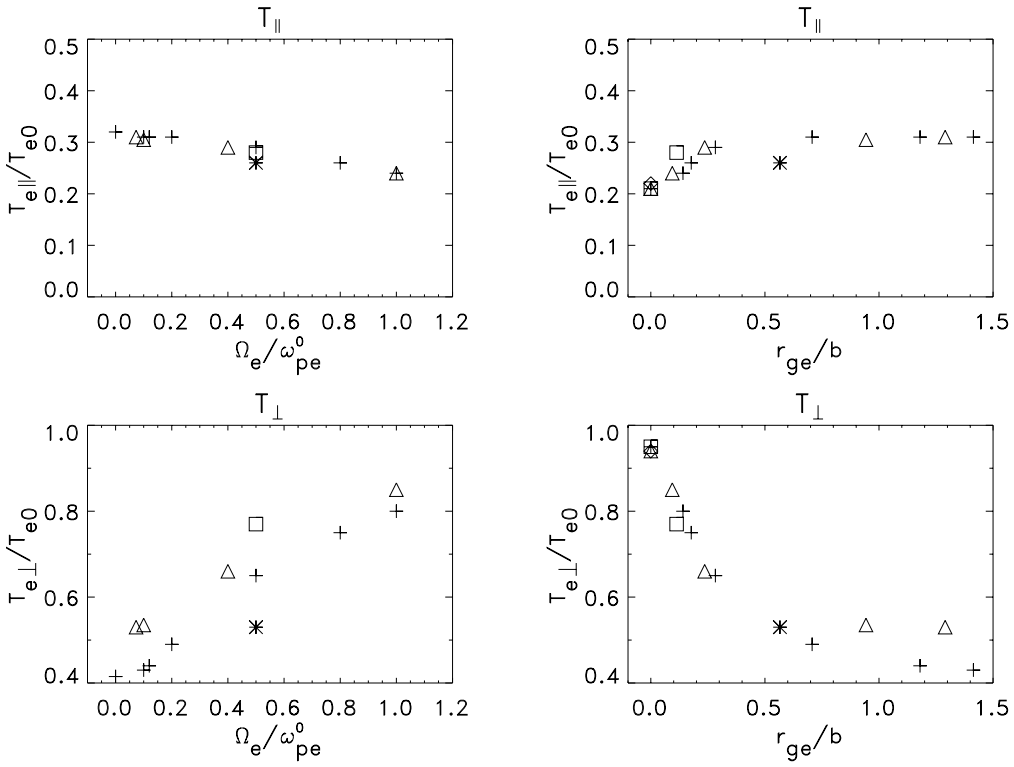

Figure 4.27: Dependence of the downstream parallel and perpendicular temperatures on ψ (left column) and on r_{ge}/b (right column) for $\eta = 1$ and various beam widths ($b = 5\lambda_D$: $*$, 10: $+$, 15: $\triangle$, 25: $\square$, 40: $\diamond$).

Two effects are responsible for the perpendicular downstream temperature to be generally lower than the injection value: (i) the escape of hot electrons with $v_{e\perp} > v_{e0}^{th}$ through the lateral beam surfaces, and (ii) the overall cooling of the electron plasma associated with the electron pressure driven adiabatic expansion of the plasma beam. Due to the huge ion inertia of $m_i = 250{,}000\, m_e$, the latter effect will be of minor importance only. Hence, the perpendicular temperature strongly depends on the fraction of electrons that manage to escape from the beam. This fraction, however, is effectively determined by how many electrons actually *reach* the beam surface. For gyroradii much smaller than the beam width, the gyromotion confines the vast majority of electrons within the beam, while only those that are less than a gyroradius away from the surface intersect the beam-vacuum

interface on their gyropath, are scattered by the strong potential gradient there and can eventually escape. For a rectangular beam, this "loss region" covers an area of roughly $r_{ge}b$, which constitutes a fraction of $r_{ge}b/b^2 = r_{ge}/b$ of the total beam cross section. This explains the strong decrease of $T_{e\perp}$ with increasing r_{ge}/b as manifested in Fig. 4.27.

For gyroradii with $r_{ge}/b \approx 1$, the loss region already covers the whole beam. Hence, the simple dependence of $T_{e\perp}$ on r_{ge}/b as just described breaks down for $r_{ge}/b > 1$, and the perpendicular temperature becomes a more complicated function of b, with wider beams having generally a higher $T_{e\perp}$.

Analogously to $T_{e\perp}$, the reason for $T_{e\|}$ to be lower downstream than upon injection, is the escape of electrons with high field-aligned velocities $v_x > v_{e0}^{th}$ through the axial beam surface, i. e. the head of the ion beam. As the axial magnetic field does not affect the electron movement in this direction, the fraction of electrons leaving the beam through the front end should be independent of the field strength. Nevertheless, the downstream parallel temperature decreases with increasing B (Fig. 4.27). This moderate decrease can be attributed to a second-order effect: The selective loss of electrons with high v_x through the head of the ion beam as the main mechanism to cause a decrease in $T_{e\|}$ becomes increasingly important when the loss through the lateral surfaces, which does not specifically select the electrons with high v_x, ceases. In other words, a strong magnetic field refrains the electrons from leaving through the lateral surfaces and thus keeps more electrons inside the beam, while the number of escaping electrons in the axial direction is unchanged, which then results in a lower $T_{e\|}$.

The overall range of $T_{e\|}$ reachable by applying magnetic fields of various strengths corresponds to a downstream range of the parallel electron thermal velocity from $0.46v_{i0}$ to $0.57v_{i0}$. This velocity enters the necessary condition for shock-like neutralization Eqn. (4.6). Taking e. g. a magnetic field of $\psi = 1$, this condition requires $\eta = v_{e0}^{th}/v_{i0} < 2$, as compared to $\eta < 1.7$ for $B = 0$. Considering that current ion thrusters are operating at $\eta \gg 1$, even such a strong magnetic field, which would correspond to 50 times the DS1 magnetic field, does not yield a major relaxation of the necessary condition for shock-like neutralization.

Moreover, as can be seen from Fig. 4.27, the temperature anisotropy rises considerably for increasing magnetic fields. A strong anisotropy represents a reservoir of free energy that might drive unwanted instabilities on time scales longer than those simulated here. Although this issue has to be investigated in more detail, in this respect, the intentional application of a strong magnetic field does not appear as an appropriate means to achieve an optimal neutralization.

In terms of the degree of non-neutrality, however, a magnetic field is very effective, as can be seen from Fig. 4.28, which shows the downstream electron density versus r_{ge}/b for the simulation runs of Fig. 4.27. For magnetic field strengths corresponding to r_{ge}/b of less than 0.3, the electron density exhibits a strong increase towards complete neutralization. This is especially interesting for narrow beams

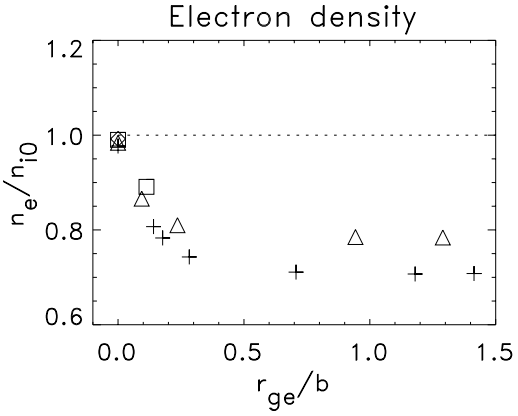

Figure 4.28: Dependence of the downstream electron density on r_{ge}/b for $\eta = 1$ and $b = 10\lambda_D$ (+), $15\lambda_D$ (△), and $25\lambda_D$ (□). The dotted line represents complete neutralization.

of a few λ_D, which turned out to be highly non-neutral in the absence of a magnetic field ($n_e/n_{i0} \approx 0.7$ for $b = 10\lambda_D$, cf. Fig. 4.22). However, the narrower the beam, the higher is the required absolute magnetic field to achieve a ratio r_{ge}/b of less than 0.3. For a beam of $b = 10\lambda_D$ with a density similar to the DS1 value, $r_{ge}/b = 0.3$ would correspond to a magnetic field strength of 5×10^{-3} T, i. e. ten times the DS1 magnetic field.

4.2.3 Technical considerations

As already outlined in Sec. 4.1.4, the shock-like neutralization scenario is of technical interest: It generates a fully thermalized downstream plasma with practically no free energy to drive instabilities and its potential structure reduces the number of backstreaming, eroding CEX ions. Similar to our simulation results in the absence of a magnetic field (Sec. 4.1.1), we showed that also for $B \neq 0$, shock-like neutralization can be enforced by suitably choosing the injection velocity ratio $\eta = v_{e0}^{th}/v_{i0}$. While in the case of an infinite magnetic field the electrons did not adopt a Maxwellian velocity distribution downstream for $\eta > 2.2$, injection velocity ratios of $\eta < 2.2$ resulted in a fully thermalized electron plasma. However, in contrast to the $B = 0$ case, this thermalization does not take place by virtue of a single electrostatic shock front, but rather via a non-linear, Landau-damped electrostatic wave. This essentially one-dimensional wave was identified as being excited by the difference between electron and ion bulk velocity upon injection.

The lateral beam dimension, which strongly affected the shock velocity v_{sh} in the absence of a magnetic field, was found to have practically no impact on the overall beam behaviour in the presence of a strong axial magnetic field corresponding to $\psi = 10^9$. Considering that the magnetic field enforces almost per-

fect space charge neutralization independent of the beam width, this observation confirms our earlier hypothesis (cf. Sec. 4.1.2) that the shock velocity variation is not a direct consequence of the changing beam width, but is rather attributable to the varying degrees of non-neutrality that came along with different beam widths. Moreover, in conjunction with the previously determined dependence of v_{sh} on n_e/n_{i0} in Fig. 4.12, the shock velocity observed here of only $0.07v_{i0}$ for $n_e/n_{i0} = 0.98$ corroborates that the shock indeed tends to stationarity for *completely* neutralized beams.

Our investigations of thruster beam neutralization in the presence of magnetic fields of intermediate strengths revealed that for a significant impact of the magnetic field on the simulated quasi-1D neutralization scenario, a minimum field strength corresponding to $\Omega_e = 0.2\omega_{pe}^0$ is required. This amounts to about 10 times the DS1 value. Hence, typical spacecraft stray fields are unlikely to have an effect on the neutralization process.

We further studied the dependence of the downstream plasma state in terms of electron temperature and degree of non-neutrality on the magnetic field strength. While the perpendicular electron temperature $T_{e\perp}$ rises significantly from $0.43T_{e0}$ for $B = 0$ to $0.95T_{e0}$ for $B \to \infty$, the parallel temperature $T_{e||}$ covers only a small range and drops from $0.32T_{e0}$ to $0.21T_{e0}$. As the parallel temperature $T_{e||}$ enters the necessary condition Eqn. (4.6) for shock-like neutralization, the intentional application of strong magnetic fields is a means to generate favorable conditions for this neutralization scenario. However, due to the small range of achievable $T_{e||}$, realistic magnetic field strengths do not yield a major relaxation of Eqn. (4.6).

A crucial parameter assessing the efficiency of the neutralization process is the degree of non-neutrality within then beam. Large net residual charges result in strong electric fields, which can accelerate CEX ions to high energies and can thus enhance erosion of spacecraft surfaces [Wang and Brophy, 1995; Samanta Roy et al., 1996a, 1996b; Wang et al., 1996]. As is readily expected from its confining character, an axial magnetic field is very effective in terms of space charge neutralization, especially for beams with small lateral dimensions. The downstream electron density exhibits a sharp increase towards complete neutralization around magnetic field strengths corresponding to $r_{ge}/b = 0.3$.

4.3 Spatially separated electron and ion sources

The simulation studies carried out so far gave already some insight into the complex process of ion thruster beam neutralization and its dependence on injection velocity ratio η, beam width b and magnetic field B. A major restriction of these investigations is, however, the simplification to a quasi-one-dimensional geometry, with electrons and ions being injected through a common opening. This section is therefore dedicated to explore the effect of the spatial separation between the electron-emitting cathode and the ion source.

4.3.1 Non-zero displacements

First of all, in order to investigate the transition from a quasi-1D configuration to one with spatially separated particle sources, we employ a simulation geometry according to Fig. 4.29. Electron and ion injection areas are shifted apart in the z direction by a variable spatial displacement δ. The quasi-1D geometry of our previous studies would correspond to $\delta = 0$, and $\delta = b$, where b is the beam width, results in a complete separation of electron and ion source. Apart from the different geometry, the simulation setting is identical to the one of Sec. 4.1. A static magnetic field is not applied.

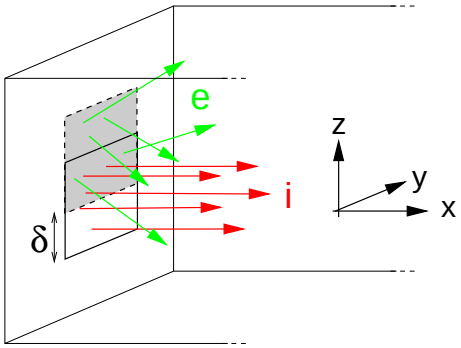

Figure 4.29: The simulation geometry. The injection areas of ions (solid line) and electrons (shaded area, dashed line) have a spatial displacement δ in the z direction.

Fig. 4.30 shows the potential and the electron density in axial direction of a beam with $b = 15\lambda_D$ for various displacements δ after $\omega_{pe}^0 t = 120$. The electrons and ions are injected at $x = 10$ with an injection velocity ratio of $\eta = 1$. At the time shown, the ion beam has moved up to $x = 160$. The potential was obtained by averaging the axial component of the electric field E_x across the *ion* beam, integrating in axial direction and setting $\Phi = 0$ in the injection plane. It is normalized to $\Phi_0 := k_B T_{e0}/e$. The electron density is also averaged across the ion beam, and is normalized in the usual manner.

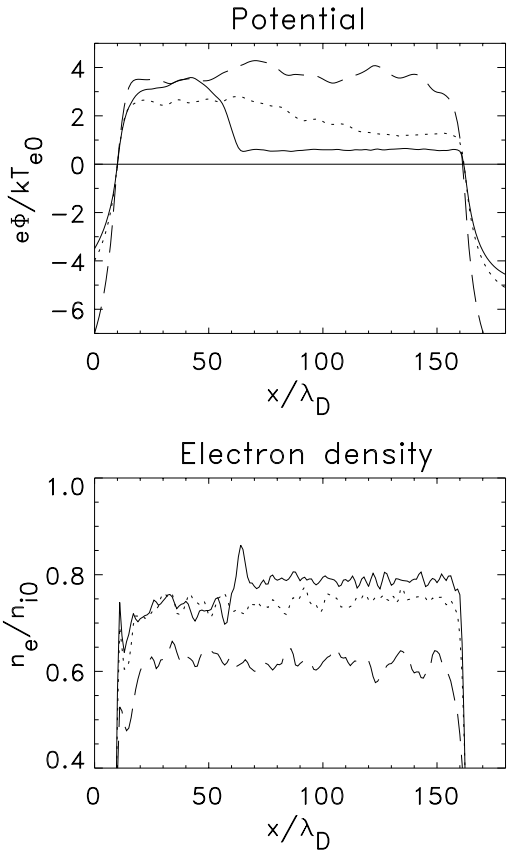

Figure 4.30: Potential and electron density in axial direction for spatial displacements between electron and ion source of $\delta = 0$ (solid line), $\delta/\mathrm{b} = 2/15$ (dotted line), and $\delta/\mathrm{b} = 5/15$ (dashed line). Beam width and injection velocity ratio were fixed at $\mathrm{b} = 15\lambda_\mathrm{D}$ and $\eta = 1$, respectively.

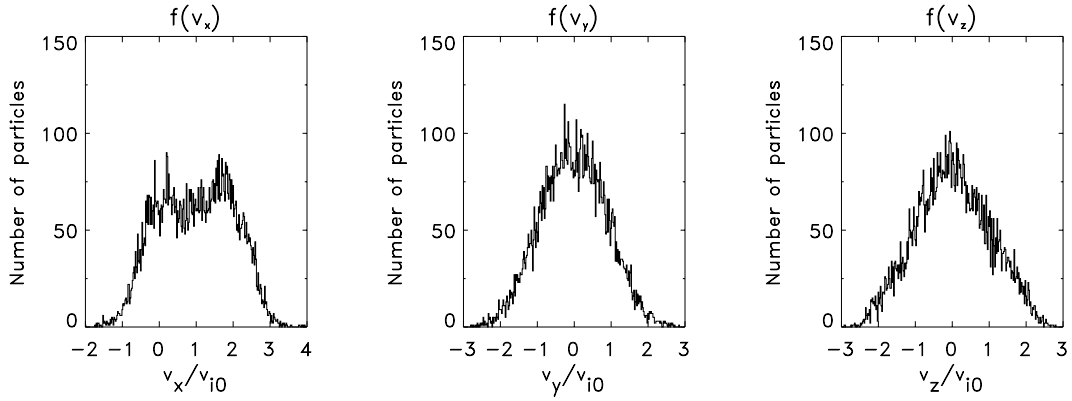

Figure 4.31: The three components of the electron velocity distribution function at $\mathrm{x} = 140\lambda_\mathrm{D}$ for $\delta/\mathrm{b} = 5/15$, $\mathrm{b} = 15\lambda_\mathrm{D}$ and $\eta = 1$.

The solid line in the top panel of Fig. 4.30 shows the well-known potential profile for the quasi-1D geometry of $\delta = 0$, as discussed in Sec. 4.1.1. Its most remarkable feature is the sharp potential drop from $3\Phi_0$ to $0.6\Phi_0$ roughly within one electron inertia length around $x = 60$. This drop was identified as a moving electrostatic shock, which generates a fully thermalized electron plasma in the adjacent region of constant potential.

As can be seen from Fig. 4.30, the initially sharp potential drop becomes weaker and finally vanishes when the displacement δ is gradually increased: For $\delta/b = 2/15$ (dotted line) the potential profile still exhibits a characteristic decrease from $x = 65$ towards a region of constant potential downstream of $x = 130$, which can be understood as a broadening of the initially sharp potential drop. If, however, δ is further increased to $\delta/b = 5/15$ (dashed line), no such feature can be identified. In this case, the potential rather fluctuates around a constant level of $3.5\Phi_0$.

In the density profile, the shock of the $\delta = 0$ case manifests itself as a jump in n_e from 0.75 to $0.8n_{i0}$ (solid line in the bottom panel of Fig. 4.30). In accordance with the disappearance of a shock signature in the potential profile upon increasing δ/b, this jump vanishes for $\delta/b = 2/15$ and $\delta/b = 5/15$. Moreover, the downstream electron density within the ion beam drops from $0.8n_{i0}$ for $\delta = 0$ to roughly $0.6n_{i0}$ in the case $\delta/b = 5/15$. This is a direct consequence of the electrons being injected farther off from the centre of the ion beam: For increasing displacements δ, the electrons are born with a higher average potential energy with respect to the ion beam. As the electron thermal velocity upon injection and thus the average kinetic energy is not changed for the different simulation runs, a greater δ causes more electrons to have enough total energy to escape from the attractive potential of the ion beam.

The downstream electron plasma in the quasi-1D configuration was found to be fully thermalized, i. e. it adopted a Maxwellian velocity distribution in all three components (cf. Sec. 4.1.1). When a spatial displacement between electron and ion source is introduced, this is no longer the case. As shown in Fig. 4.31, which displays the electron velocity distribution roughly 10 electron inertia lengths downstream of the injection plane for $\delta/b = 5/15$, it departs significantly from a thermalized distribution. Only v_y still exhibits a Maxwellian shape. Since the spatial shift δ is in the z direction, i. e. perpendicular to y, it was to be expected that v_y is the least affected component.

The z component of the electron velocity rather exhibits a triangularly shaped distribution. As the electrons are fully Maxwellian in this component upon injection, this is not due to the absence of the electrostatic shock that provides thermalization in the quasi-1D case, but a pure effect of the spatial displacement δ.

In the axial velocity component v_x, the electrons are injected with a *half*-Maxwellian velocity distribution, i. e. in this component they do not enter the simulation box in thermal equilibrium. A thermalization of the electrons in v_x can take place either by virtue of the electrostatic shock of the $\delta = 0$ case, or, in a more

continuous manner, by some other kind of wave-particle interaction. The latter option would not lead to a complete thermalization within a few electron inertia lengths behind the injection plane, but would require a greater distance (Sec. 4.1.1). Hence, the fact that the electrons do not thermalize in v_x within the observed spatial scale is directly related to the disappearance of the shock.

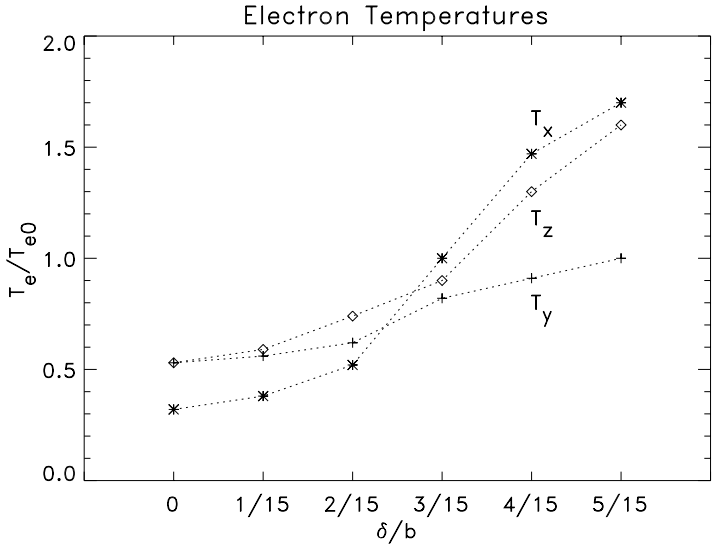

Figure 4.32: Dependence of the electron temperatures on the spatial displacement δ for $\mathsf{b} = 15\lambda_\mathsf{D}$ and $\eta = 1$. For the non-Maxwellian distributions beyond $\delta/\mathsf{b} = 2/15$, the temperatures are determined as the standard deviations of the respective velocity components.

A non-zero displacement δ has also an effect on the electron temperatures. Their dependence on δ/b is shown in Fig. 4.32. For the non-Maxwellian distributions beyond $\delta/b = 2/15$ we took the standard deviations of the respective velocity components as T_x, T_y, and T_z. While all three temperatures rise significantly with increasing δ, it is again the y value that is the least affected. The general trend towards higher temperatures for increasing displacements δ can be understood as a consequence of the enhanced potential energy, with which the electrons are born upon injection: Those electrons that do not have enough total energy to escape from the attractive potential of the ions are captured by the beam, thereby transforming their potential energy into kinetic energy. The farther off from the ion beam centre the electrons are injected, the higher is the kinetic energy, i. e. the temperature, with which they end up once captured within the ion beam.

The quasi-1D studies of the previous sections identified the downstream electron temperature as the crucial parameter determining whether a thermalizing electrostatic shock front builds up or not. As a necessary condition for shock-like neutralization we found that the electron thermal velocity on the downstream

side v_e^{th} has to be smaller than the ion beam velocity v_{i0}:

$$v_{i0} > v_e^{th}. \tag{4.6}$$

We were able to confirm the validity of condition (4.6) in a series of simulation runs with different injection velocity ratios $\eta = v_{e0}^{th}/v_{i0}$ (Sec. 4.1.1). It was shown to hold also in the case of a static magnetic field applied in the axial direction (Sec. 4.2.1). Considering now that $v_e^{th} = \sqrt{k_B T_e/m_e}$ and that $\eta = v_{e0}^{th}/v_{i0} = 1$ for the simulation runs of Fig. 4.32, it can be seen that around $\delta/b = 3/15$ the necessary condition for shock-like neutralization (4.6) starts to be violated. Therefore, one might wonder if this violation is the actual reason for the disappearance of the electrostatic shock for $\delta/b \gtrsim 3/15$ and if the shock-like neutralization regime can be recovered by choosing the injection velocity ratio η such that v_{i0} exceeds v_e^{th} downstream and thus fulfills Eqn. (4.6).

In order to answer this question, we have carried out a simulation run with $\delta/b = 3/15$ and $\eta = 0.5$ instead of $\eta = 1$. The potential profile, the electron phase space $v_x - x$, and the electron density within the ion beam for this run are shown in Fig. 4.33. For comparison, the corresponding results of the $\eta = 1$ case are displayed in Fig. 4.35.

There is a remarkable difference in the beam behaviour between the $\eta = 1$ and the $\eta = 0.5$ case, with the latter indeed resembling much more the shock-like neutralization scenario of the quasi-1D situation. In contrast to the rather flat potential of Fig. 4.35, the potential profile for $\eta = 0.5$ exhibits a strong decrease around $x = 180$ towards a region of roughly constant potential. This is a characteristic feature of the quasi-1D shock-like neutralization (Secs. 4.1.1 and 4.2.1). The spatial extent of this transition region is, however, quite different. For $\delta/b = 3/15$, it covers a region of about $60\lambda_D$, as compared to $10 - 15\lambda_D$ for $\delta = 0$.

While the electron phase space of the $\eta = 1$ case in Fig. 4.35 does not change much in axial direction, it undergoes a significant transition around the potential decrease for $\eta = 0.5$. Similar to what we observed in the quasi-1D configuration, the electron phase space on the upstream side consists of a beam component around $1.8v_{i0}$ and a trapped hot component. Downstream of the potential decrease, the electrons adopt a more symmetric distribution. However, as can be seen from Fig. 4.34, they do not become Maxwellian in all three components, as they do for $\delta = 0$. As already observed in the $\eta = 1$ case (Fig. 4.31), the distribution in v_z rather exhibits a triangular shape, while v_y is again the component whose distribution comes closest to a Maxwellian.

Hence, we note that despite leading to a potential profile and to electron phase space features that are reminiscent of the quasi-1D shock-like neutralization scenario, the fulfilment of Eqn. (4.6) with a suitably chosen η does not generate a fully thermalized downstream plasma in the case of non-zero displacements δ. This underlines the character of Eqn. (4.6) as being only a *necessary* and not *sufficient* condition for shock-like neutralization.

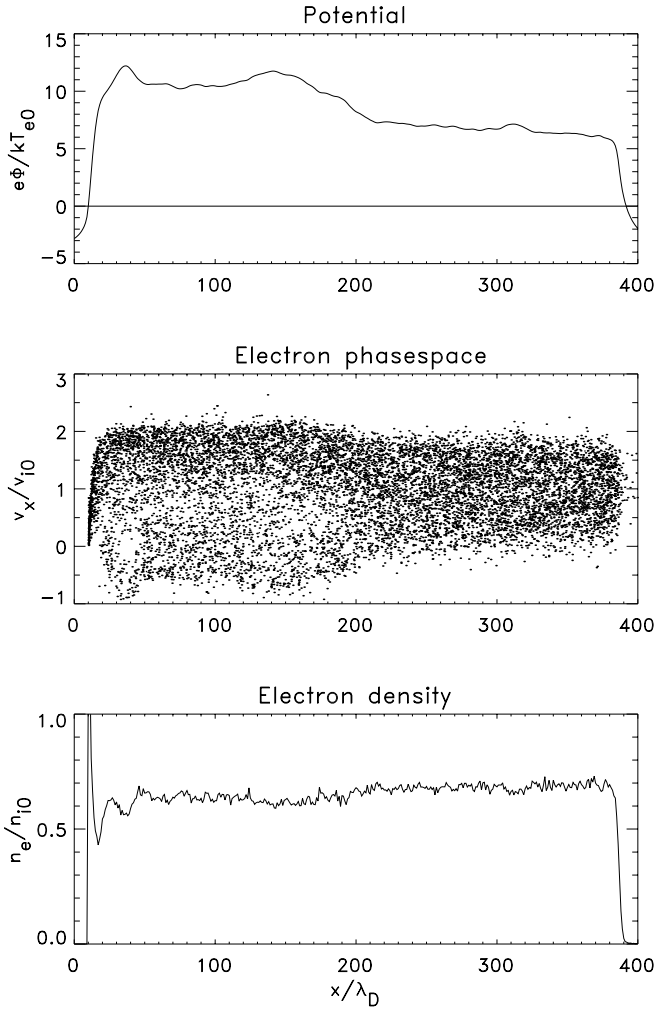

Figure 4.33: Potential, electron phase space $v_x - x$, and electron density for $\eta = 0.5$, $\delta/b = 3/15$, and $b = 15\lambda_D$.

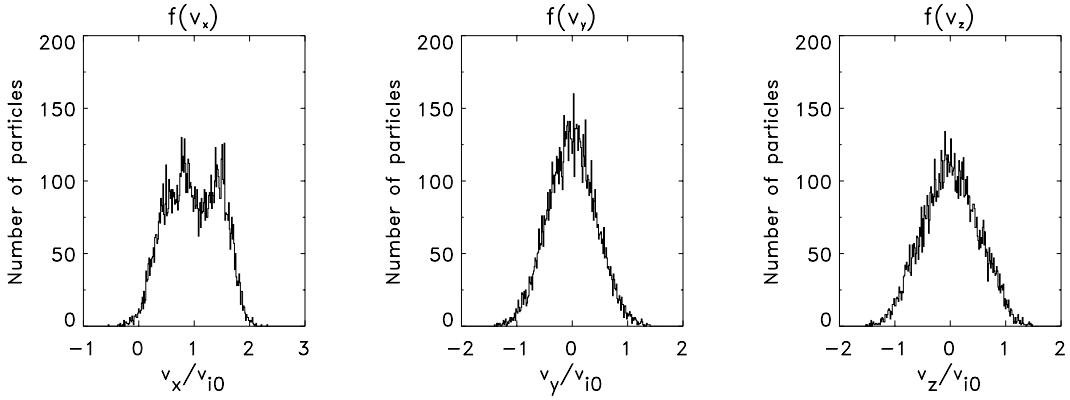

Figure 4.34: The three components of the electron velocity distribution function at $x = 320\lambda_D$ for the simulation run of Fig. 4.33.

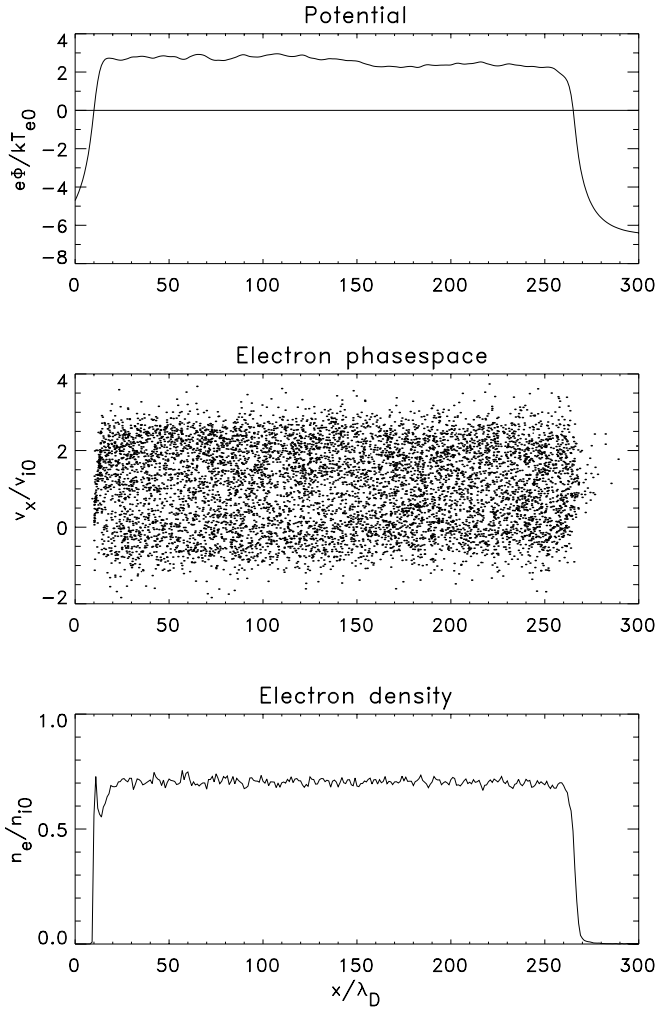

Figure 4.35: Same as Fig. 4.33, but with $\eta = 1$ instead of $\eta = 0.5$.

4.3.2 Complete separation of electron and ion sources

In this section, the plasma dynamics for the case of a complete separation between the particle sources are examined. The injection areas of both particle species do not overlap anymore, and – in order to get closer to the actual geometry of existing ion thrusters – they are now round and of different size. Their diameters are fixed at $3\lambda_D$ for the electrons and at $15\lambda_D$ for the ions (Fig. 4.36).

Figs. 4.37 to 4.42 show the results of a simulation run in such a configuration for the case $\eta = 1$. As usual, the potential is computed by integrating the electric field in the axial direction, averaging across the ion beam, and setting $\Phi = 0$ in the injection plane (Fig. 4.37). The electron density, however, is obtained in a different way than it was before: In the quasi-1D simulations so far and also for slight displacements δ, the electron density outside the ion beam was close to zero. Therefore, the average electron densities appearing in the plots were

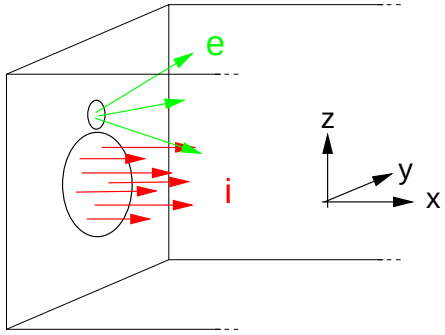

Figure 4.36: The simulation geometry with completely separated, round injection areas.

calculated by simply counting the electrons *within the ion beam*. With the electron source being now outside the ion beam, this density is not a good measure of the overall degree of neutralization anymore, because a significant fraction of electrons contributing to the neutralization actually reside *around* the beam. More appropriate is the "slice density", which counts all electrons in a given slice of the simulation domain. Hence, from now on, the plotted electron densities are such slice densities (normalized to the nominal ion slice density of a non-diverging beam n_{i0}).

As expected from our observations in the previous section, shifting the electron source completely outside the ion beam involves an enhanced loss of electrons. Their average density drops to $0.45n_{i0}$ (Fig. 4.37), whereas in the corresponding quasi-1D case it was around $0.8n_{i0}$ (Fig. 4.12). As a consequence of the high degree of non-neutrality, the beam potential rises to $24\Phi_0$ with respect to the injection plane and to around $52\Phi_0$ with respect to the ambient level (Fig. 4.37), as compared to $4\Phi_0$ and $10\Phi_0$ for the quasi-1D case.

The structure of the potential is that of a reversed potential trough: Behind a sharp increase within the injection sheath of roughly $20\lambda_D$ thickness, it remains flat up to the front end of the ion beam where it drops back to the ambient level. That there is no sign of an electrostatic shock anymore does actually not surprise: With an electron temperature of around $5T_{e0}$ (see further down), the necessary condition Eqn. (4.6) for the occurrence of the shock is not fulfilled.

Despite the strong degree of non-neutrality, there is practically no axial electric field within the beam. Similar to previous observations, the electrons arrange themselves in such a manner that the whole electric flux associated with the vast excess of positive charges within the beam emanates through the lateral surfaces.

One of the basic questions of ion beam neutralization addresses the adaptation of the average electron velocity to the ion bulk movement in the absence of collisions. The electron phase space in Fig. 4.37 gives a clue on how this is accomplished: Within the injection sheath, the electrons are accelerated to form a beam

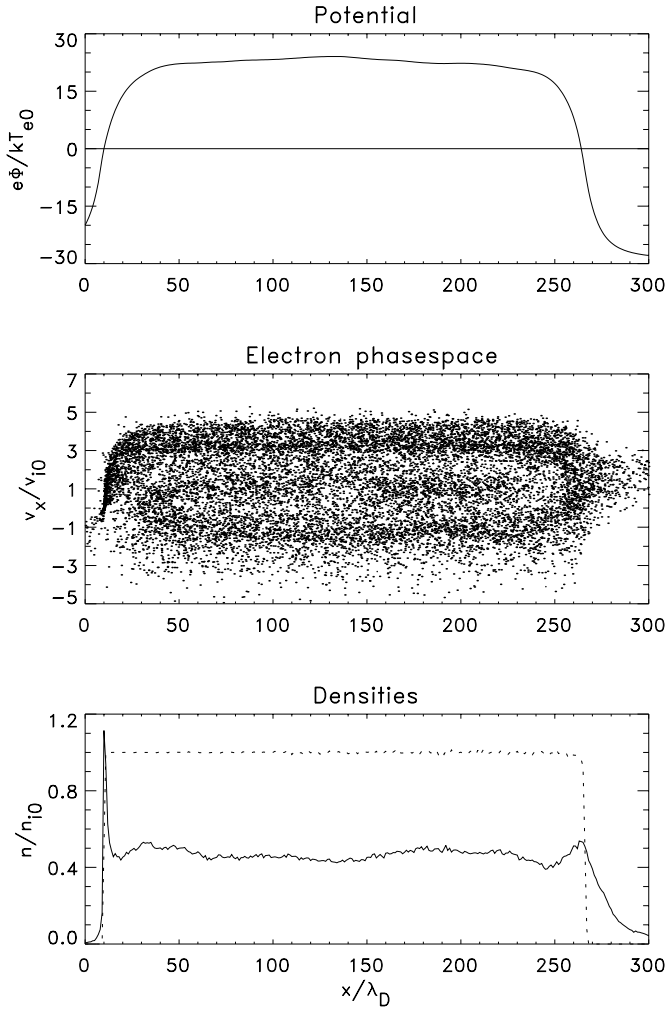

Figure 4.37: Potential, electron phase space $v_x - x$, and slice densities of electrons (solid) and ions (dotted) for a configuration according to Fig. 4.36 with $\eta = 1.0$ and $b = 15\lambda_D$.

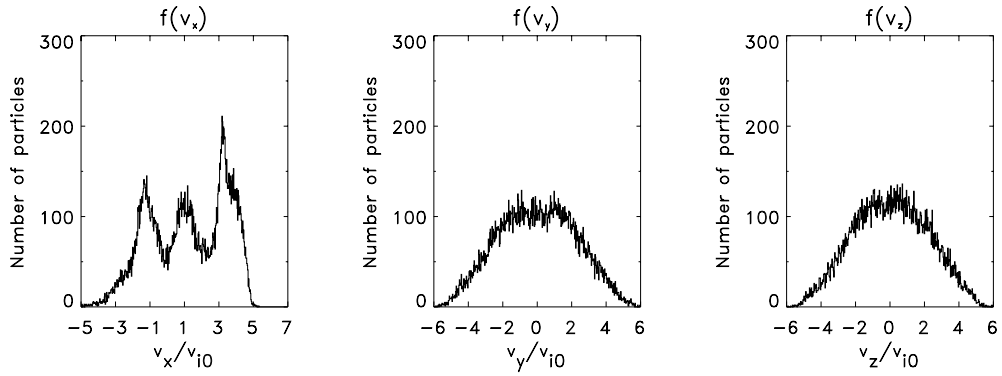

Figure 4.38: The three components of the electron velocity distribution function at $x = 220\lambda_D$ for the simulation run of Fig. 4.37.

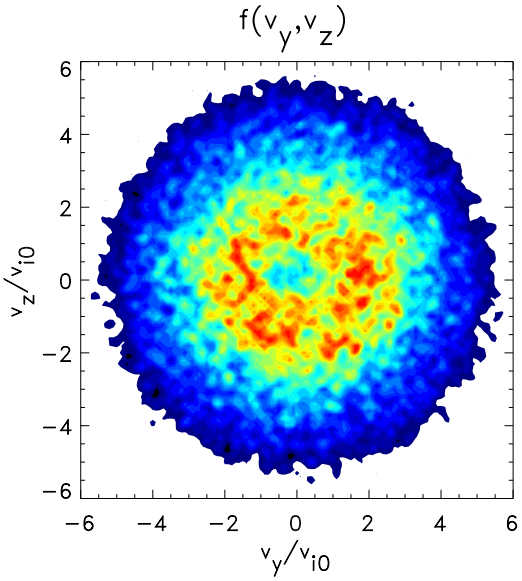

Figure 4.39: The electron distribution function of the perpendicular velocity components v_y and v_z in the region between $x = 180\lambda_D$ and $x = 220\lambda_D$. Hot and cold colors correspond to great and small numbers of particles, respectively.

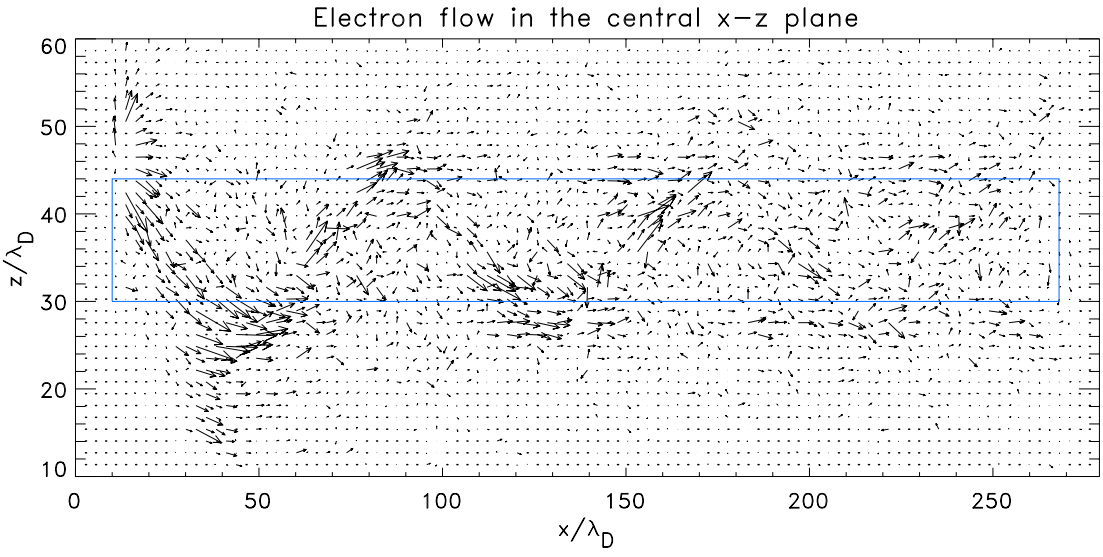

Figure 4.40: The electron mass flow in the central x-z plane for the simulation run of Fig. 4.37. The blue line marks the extent of the ion beam.

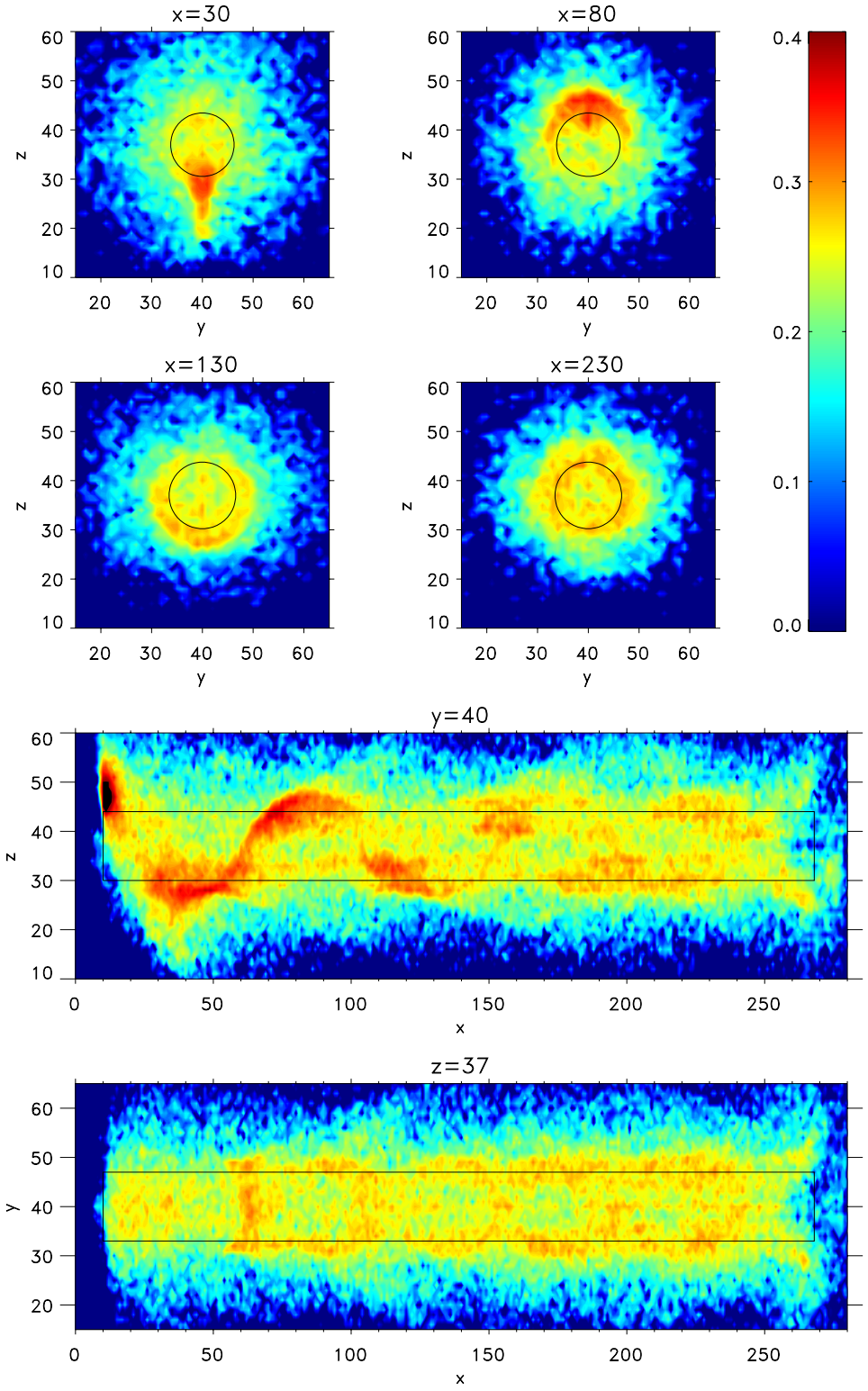

Figure 4.41: Electron density cuts for the simulation run of Fig. 4.37 ($\eta = 1$). The coordinates are given in units of λ_D, and the density values are normalized to n_{i0}. The black lines indicate the extent of the ion beam. Around the injection plane, the electron density exceeds the applied color scale and reaches values of up to $1.2 n_{i0}$.

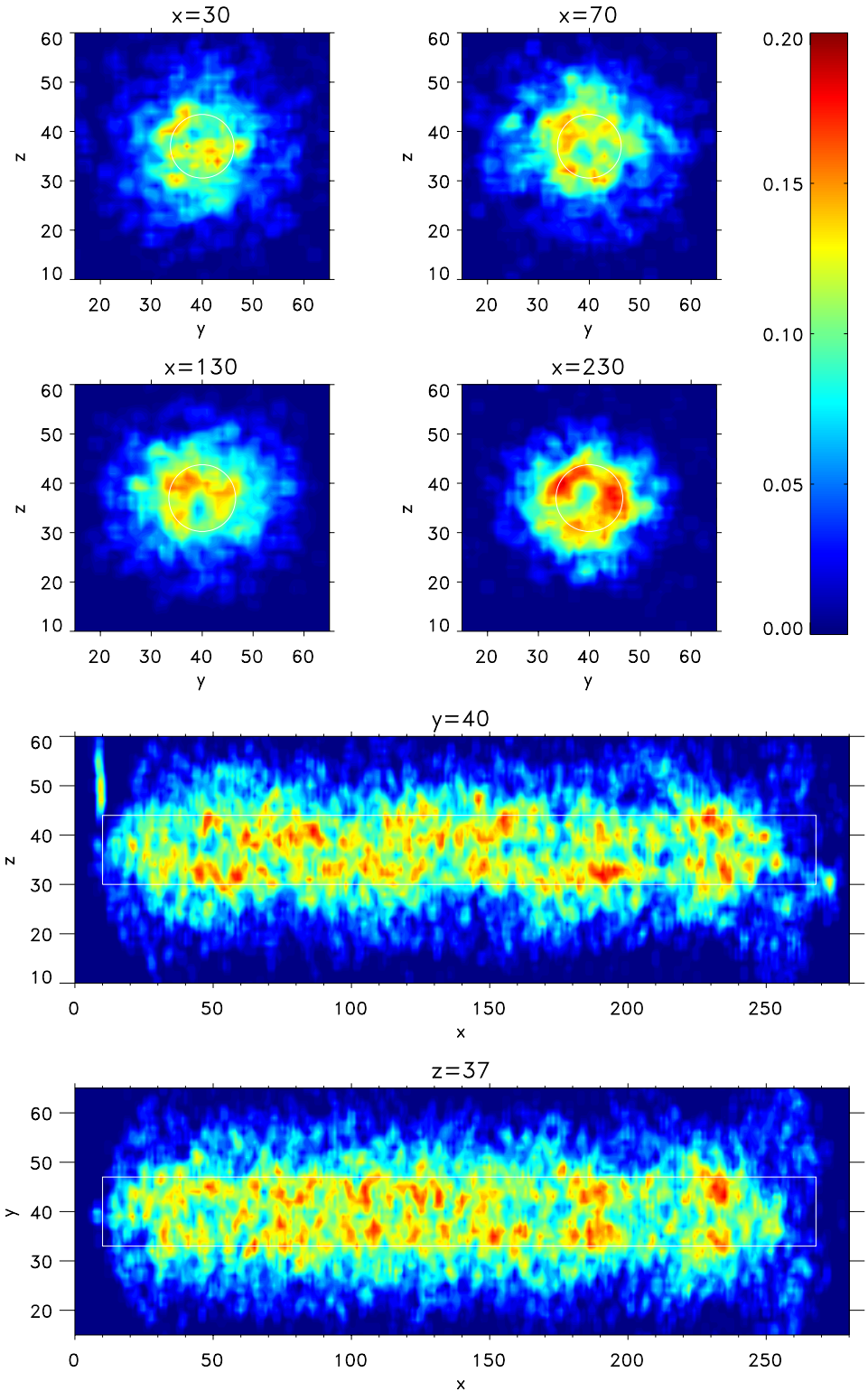

Figure 4.42: Same as Fig. 4.41, but only showing the backward streaming electrons, i. e. those with velocities v_x smaller than v_{i0}.

at about $3v_{i0}$. With this velocity, they overtake the ions and stream up to the head of the ion beam, where they are reflected. As the head itself is moving at v_{i0}, the reflected electrons acquire a velocity of $(-(3-1)+1)v_{i0} = -v_{i0}$. They travel back to the *stationary* injection sheath, where they are again reflected, and end up with a velocity of v_{i0}, i. e. they are streaming with the ion bulk velocity.

In other words, after their initial acceleration in the injection sheath, the electrons undergo a sort of "Fermi-deceleration" between the expanding ends of the ion beam, which adapts their bulk velocity to v_{i0}. In the simulation run discussed here, this adaptation mechanism seems quite fortunate: Just two reflections are needed to provide the electrons with exactly the right velocity. For other situations, where the electrons are not accelerated to roughly $3v_{i0}$ in the injection sheath, this velocity adaptation might not be so effective. More than two reflections between the stationary injection sheath and the moving head of the ion beam might be necessary, and the electrons might not end up with exactly v_{i0}. However, already after a single reflection at the head of the ion beam, the electrons streaming with v_e in the $+x$ direction acquire an *average* velocity among forward streaming and reflected electrons that exactly equals the ion bulk velocity:

$$v_{refl} = -(v_e - v_{i0}) + v_{i0} \;\Rightarrow\; v_{av} = \frac{1}{2}\left(v_e + v_{refl}\right) = v_{i0}\,. \tag{4.31}$$

Hence, although being probably not always as effective as in the run of Fig. 4.37, the quasi-Fermi-decelaration represents a universal mechanism of adapting the average electron velocity to the ion bulk movement in the absence of collisions; i. e. as long as the distance traveled by the electrons does not significantly exceed their mean free path, which was determined to be in the order of 24 m in the case of DS1 (Sec. 2.4). Later on, the reflection of electrons at the head of the ion beam becomes increasingly unlikely, since electron-ion collisions will tend to diminish any velocity difference between the two particle species. The Fermi-deceleration will then become unimportant as a velocity adaptation mechanism and will be replaced by particle collisions, which will provide a complete thermalization of the plasma.

The three distinct electron beams at $v_x = -v_{i0}$, v_{i0}, and $3v_{i0}$ show up clearly in the velocity distribution function (Fig. 4.38). In contrast to the results of slight source displacements δ, v_y can no longer be distinguished as the component that comes closest to a Maxwellian distribution, and v_z does not have a triangular shape anymore. The distributions of v_y and v_z are quite similar and rather represent flattened Maxwellians. The standard deviation of all three velocity components is roughly $5T_{e0}$, i. e. the electrons here are considerably "hotter" than for the slight displacements in the previous section. The reason for the increase in kinetic energy is again the enhanced potential energy of the electrons upon injection that is associated with the outward shifting of the electron source.

Fig. 4.41 gives an impression of the three-dimensional behaviour of the electrons. It shows contour plots of the electron density for the simulation run of Fig. 4.37 in

four cross sections at $x = 30$, 80, 130, and $230\lambda_D$ as well as in the $x - z$ plane and the $x-y$ plane. The black lines roughly indicate the extent of the ion beam, which, as a consequence of using the actual electron to ion mass ratio of 1:250,000, shows only a slight divergence.

The highest electron density is encountered in front of the electron source, where it amounts to about $1.2n_{i0}$ (see also Fig. 4.37). From there, the electrons are accelerated towards the ion beam centre by the attractive potential of the ions. This takes place within the first $10 - 20\lambda_D$ in axial direction ("injection sheath"). The electrons overshoot and build up a density enhancement on the opposite side of the beam with a maximum density of about $0.35n_{i0}$. From this region, they are accelerated back towards the beam centre to follow a meandering path between the top and bottom surfaces of the ion beam (see also Fig. 4.40).

Up to around $x = 70\lambda_D$ or $5l_e$, this path exhibits quite a coherent structure, with the electron density being localized in the central $x - y$ plane. Further downstream, it becomes more diffuse and the electrons gradually distribute over the whole ion beam circumference. As can clearly be seen from Fig. 4.41, the electrons do not fill the beam homogeneously. They concentrate around the ion beam surface, while the centre is practically void of electrons. It is the high kinetic energy gained after falling from the injection plane into the potential trough of the ion beam that makes the electrons circulate around the beam rather than reside in its centre. Accordingly, the electron distribution function in v_y and v_z resembles a ring distribution (Fig. 4.39). This also explains why the one-dimensional distributions of v_y and v_z looked like *flattened* Maxwellians (Fig. 4.38).

The meandering path is followed by the greatest part of the injected electrons. Once they are reflected at the head of the ion beam, their movement back to the injection sheath is not so structured anymore. Although forming a clearly distinguishable beam in phase space (Fig. 4.37), in real space, the reflected electrons are rather randomly distributed (Fig. 4.42). They still tend to avoid the beam centre, though.

The wavelength of the meandering path is in the order of $80\lambda_D$ or $5.7l_e$. It can be understood as the superposition of a lateral electron oscillation around the positively charged beam with the electron movement in the $+x$ direction: We consider a single electron at rest on the border of an infinitely long, positively charged cylinder. The electric field of such a cylinder charge can readily be calculated by integrating Gauss' law

$$\int \mathbf{E} \cdot d\mathbf{A} = Q/\varepsilon_0 \,, \tag{4.32}$$

which – assuming cylindrical symmetry – yields

$$E(r) = \begin{cases} \dfrac{\rho}{2\varepsilon_0} \cdot r & \text{for } r \le R \\ \dfrac{\rho}{2\varepsilon_0} \cdot \dfrac{R^2}{r} & \text{for } r > R \,, \end{cases} \tag{4.33}$$

where R is the cylinder radius and ρ is the charge density inside the cylinder. For $r \leq R$, the force on the electron is

$$F(r) = -\frac{e\rho}{2\varepsilon_0} r =: -Dr\,. \tag{4.34}$$

Hence, the electron will carry out a harmonic oscillation around $r = 0$ with a frequency

$$\omega = \sqrt{\frac{D}{m_e}} = \sqrt{\frac{e\rho}{2\varepsilon_0 m_e}}\,. \tag{4.35}$$

Introducing the electron plasma frequency of a completely neutralized cylinder

$$\omega_{pe}^0 = \sqrt{\frac{e^2 n_{i0}}{\varepsilon_0 m_e}} \tag{4.36}$$

and considering that the charge density of the cylinder is

$$\rho = e(n_{i0} - n_e)\,, \tag{4.37}$$

where n_{i0} and n_e are the actual ion and electron densities inside the cylinder, the oscillation frequency can be written as

$$\omega = \omega_{pe}^0 \sqrt{\frac{1}{2}\left(1 - \frac{n_e}{n_{i0}}\right)}\,. \tag{4.38}$$

The wavelength of the oscillation in the x direction is

$$\lambda = \tau\, v_x = \frac{2\pi}{\omega}\, v_x\,, \tag{4.39}$$

with τ being the oscillation period and v_x being the axial velocity of the injected electrons. We normalize v_x to the electron thermal velocity upon injection

$$\hat{v} := v_x / v_{e0}^{th} \tag{4.40}$$

and make use of

$$\omega_{pe}^0 = \frac{1}{\sqrt{2}} \frac{v_{e0}^{th}}{\lambda_D} \tag{4.41}$$

to obtain

$$\lambda = \frac{4\pi\hat{v}}{\sqrt{1 - n_e/n_{i0}}} \cdot \lambda_D\,. \tag{4.42}$$

The wavelength of the electron oscillation as measured in λ_D depends on the normalized streaming velocity of the injected electrons $\hat{v}$ and the degree of non-neutrality $1 - n_e/n_{i0}$ *inside* the ion beam.

For the simulation run discussed here, $\hat{v}$ can be taken from Fig. 4.38 to be around $\hat{v} = 3 \ldots 5$. In order to compute the degree of non-neutrality inside the beam,

the "slice density" of Fig. 4.37 is now not the appropriate electron density. We rather have to take the electron density within the ion beam, which was obtained to be around $n_e \approx 0.25 n_{i0}$ (not shown), giving a degree of non-neutrality of $1 - n_e/n_{i0} \approx 0.75$. Substituting these values into Eqn. (4.42) yields a wavelength of about $58\lambda_D$, as compared to $80\lambda_D$ in the simulation. Considering that Eqn. (4.42) for the oscillation wavelength is based on a rather simple model, this correspondence is quite good and confirms that the meandering electron path can indeed be understood as the superposition of the axial electron movement with an oscillation around the positively charged beam.

4.3.3 Dependence on velocity ratio η

For quasi-1D configurations and non-zero displacements between electron and ion source, the injection velocity ratio $\eta = v_{e0}^{th}/v_{i0}$ was identified as a crucial parameter for the overall beam behaviour. With a suitably chosen η according to Eqn. (4.6), the thruster beam could be made to undergo a shock-like neutralization process. In the case of spatially separated electron and ion sources, the high electron temperatures in the order of $5T_{e0}$ (cf. Sec. 4.3.2) would require an η of less than $1/\sqrt{5} \approx 0.4$. This is, however, below the presently accessible range of our code. For the moment, we therefore have to restrict our investigations of beam neutralization with spatially separated particle sources to the shock-free regime.

In order to assess the impact of η on the shock-free neutralization process in the case of fully separated particle sources, we carried out three simulation runs similar to the one of the preceding section (from now on referred to as "run 1"), but with $\eta = 2, 4, 8$ (run 2, 3, and 4) instead of $\eta = 1$. As usual, the different injection velocity ratios were realized by keeping the electron thermal velocity fixed at $0.1c$ and varying v_{i0} accordingly. The total simulation time of each run was chosen such that the ion beam reaches roughly the same spatial extent by the end of the run. This lead to simulation times of 30, 60, 120, and 240 electron plasma periods, respectively. We note that in order to keep the ion *density* constant, we adapted the number of injected particles per time step according to η. The results of these simulation runs are displayed on pages 140 to 154.

The electron densities shown in Figs. 4.43, 4.48, and 4.53 are again the "slice densities". They rise from $0.45 n_{i0}$ for $\eta = 1$ to $0.82 n_{i0}$ for $\eta = 8$, meaning that for increasing η, the number of electrons with enough total energy to escape from the beam is decreasing significantly. The reason for this decrease can be found in Fig. 4.62, which shows the electric potential along z in the centre of the injection plane: For low η, the potential exhibits a dip around the electron injection area, which vanishes for increasing η. Hence, the electrons in run 1 are born with a much larger potential energy with respect to the ambient level than those of run 4.

The potential dip itself is related to the electron density in front of the electron injection area: For all η, the ion density is kept constant by varying the particle

injection rate according to η. Since the electron injection velocity is not changed, the injection rate directly affects the density of electrons accumulating in front of the electron injection area. When reducing η, this density increases and gives rise to the significant excursion of the potential as seen in Fig. 4.62.

As a consequence of the pronounced η-dependence of the neutralization degree, also the potential within the ion beam varies strongly with η (top panels of Figs. 4.37, 4.43, 4.48, and 4.53). With respect to the ambient level, it decreases from $52\Phi_0$ in run 1 to $21\Phi_0$ in run 4, thereby following quite well a power law according to

$$\Phi/\Phi_0 = 51 \cdot \eta^{-0.43} \,. \tag{4.43}$$

Its structure is still that of a reversed potential trough (in three dimensions, see Fig. 4.61). In runs 3 and 4, however, the trough "depth" decreases along the axial direction. This is due to the decrease of the ion density in the x direction that goes along with the *lateral expansion* of the ion beam (Figs. 4.60 and 4.63). Of course, the beam also expands laterally in runs 1 and 2. Within the shorter simulation times of these runs, however, the expansion is not significant. Further down, we will explore the lateral expansion of the beam in more detail.

The phase space $v_x - x$ for runs 2-4 (middle panels of Figs. 4.43, 4.48, and 4.53) is filled much more homogeneously than for run 1, where three distinct electron beams were identifiable, and the downstream distribution function for the axial velocity component $f(v_x)$ is getting flatter for increasing η (left panels of Figs. 4.38, 4.44, 4.49, and 4.54). In all cases, despite the increasing difference between electron and ion velocity upon injection, the *average* electron velocity is roughly equal to the ion bulk velocity. In run 1, a sort of Fermi-deceleration was found to be responsible for this velocity adaptation. There is no reason to assume that this process should not be at work also in the other runs. However, it does not show up in the $v_x - x$ phase space as obviously as in run 1. In simulation run 4, we therefore traced a series of electrons from their injection up to the end. The general pattern of their movement in the $v_x - x$ phase space is similar to the one shown in Fig. 4.58. Through multiple reflections between the moving head of the ion beam and the injection sheath, the electrons indeed adapt their velocity to the ion bulk movement.

Right after its injection, the electron of Fig. 4.58 gains a maximum velocity v_x of $21v_{i0}$ and, after a total of seven reflections at the head of the ion beam, its maximum velocity is around $8v_{i0}$ (bottom panel of Fig. 4.58). This amounts to an average absolute velocity change per reflection of $1.6v_{i0}$, which is quite close to the theoretical value $2v_{i0}$ for a reflection at a wall moving at v_{i0}. This suggests that the Fermi-deceleration mechanism is at work also in the case of $\eta > 1$.

In general, the velocity change per reflection shows up in the distribution function $f(v_x)$ as the separation of electron beams that arise through multiple reflections. Hence, with a smaller v_{i0}, the electron phase space is filled much more "smoothly" than with a big v_{i0}. This explains the more homogeneous phase space

and the flatter distribution of the axial velocities for increasing η as observed in Figs. 4.38, 4.44, 4.49, and 4.54.

However, the simple Fermi mechanism with multiple reflections between a stationary wall and one moving at $2v_{i0}$ does not give the whole picture of the actual electron deceleration process: As can be seen from the bottom panel of Fig. 4.58, the electron is not reflected elastically at the injection sheath, but ususally gains some energy there, and at the front end of the ion beam it loses more than $2v_{i0}$. Moreover, among the phase space traces recorded in run 4 were also such as depicted in Fig. 4.59, where the electron transitionally *gains* kinetic energy at both ends before getting slowed down via the Fermi mechanism. Such accelerations were, however, quite rare. Hence, although giving already the right idea of the electron deceleration mechanism, our simple model cannot explain these details. They will have to be investigated in future studies.

As far as the perpendicular velocities are concerned, Figs. 4.39, 4.45, 4.50, and 4.55 show that the ring-like distribution of run 1 gradually vanishes upon decreasing v_{i0}. This is probably due to the more frequent electron reflections at the ion beam ends in the case of lower v_{i0}, which can be expected to destroy such a non-thermal phase space symmetry.

The widths of the velocity distributions, i. e. the electron temperatures, are directly related to the amount of potential energy that is transformed into kinetic energy upon falling from the injection area into the potential trough of the ion beam (cf. Sec. 4.3.1). With decreasing trough depths for growing η, also the electron temperatures decrease (Fig. 4.64). A logarithmic regression yields the following power law for the η-dependence of the electron temperature:

$$T_e/T_{e0} = 5.1 \cdot \eta^{-0.3} \,. \tag{4.44}$$

While for runs 1 and 2, the temperatures are essentially constant along the x direction, there is an appreciable axial decrease of T_e in runs 3 and 4 (Figs. 4.48 and 4.53). This is a consequence of the axial potential slope in these runs: The electrons gain potential energy at the cost of their kinetic energy.

The three-dimensional distribution of electrons (Figs. 4.41, 4.47, 4.52, and 4.57) and their flow pattern (Figs. 4.40, 4.46, 4.51, and 4.56) exhibit a remarkable development from run 1 to run 4. The very structured electron distribution of run 1, which is dominated by the meandering path, is gradually replaced by a rather homogeneous spreading of the electrons over the ion beam. Apparently, the higher frequency of electron reflections at the head of the ion beam associated with an increase of η not only provides a smoother filling of the phase space (see above), but also of the real space.

Even though it does not dominate the electron distribution anymore, the meandering path of the electrons is still visible in runs 2-4. At least 1 1/2 oscillations can be traced, before the flow field becomes more turbulent and the density of the meandering electrons disappears in the "background" electron density.

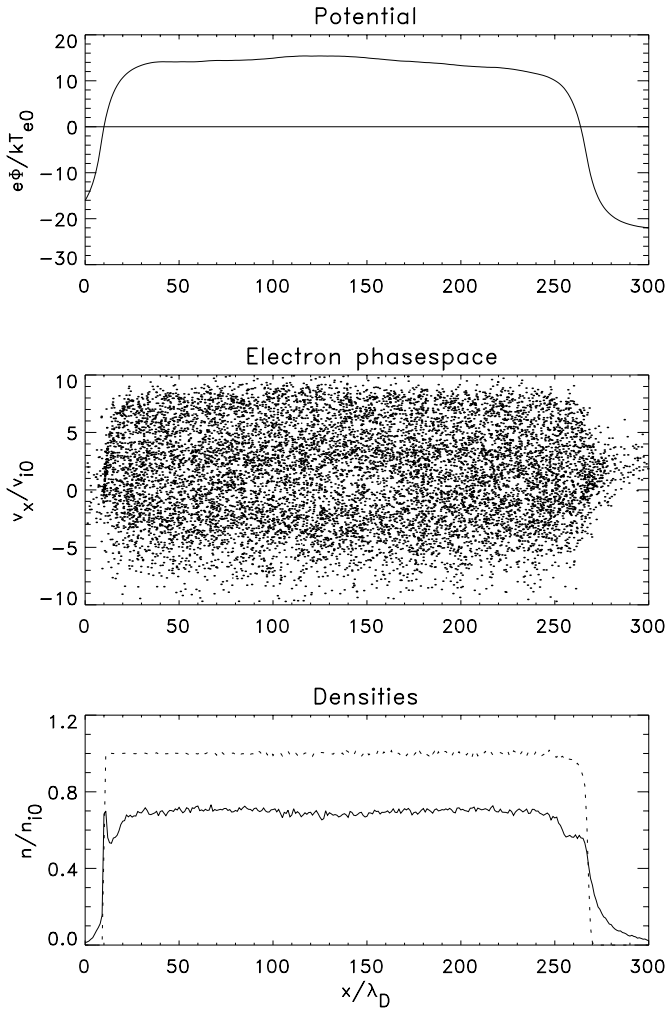

Figure 4.43: Same as Fig. 4.37 but with $\eta = 2$.

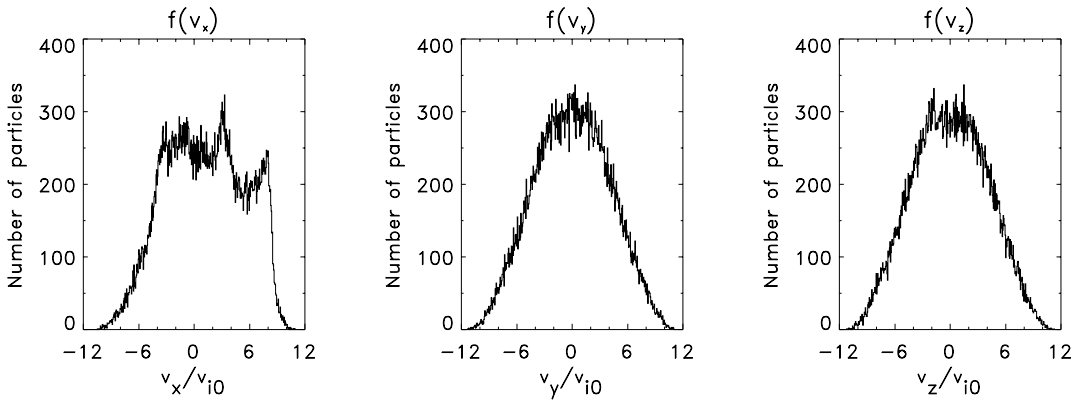

Figure 4.44: The three components of the electron velocity distribution function at $x = 220\lambda_D$ for $\eta = 2$.

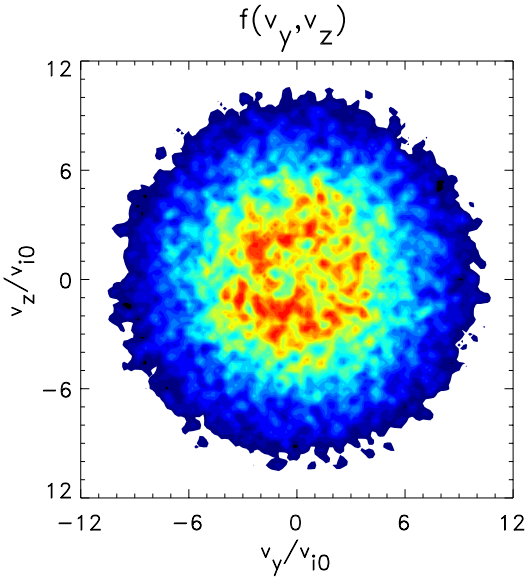

Figure 4.45: The electron distribution function of the perpendicular velocity components v_y and v_z in the region between $x = 180\lambda_D$ and $x = 220\lambda_D$ for $\eta = 2$.

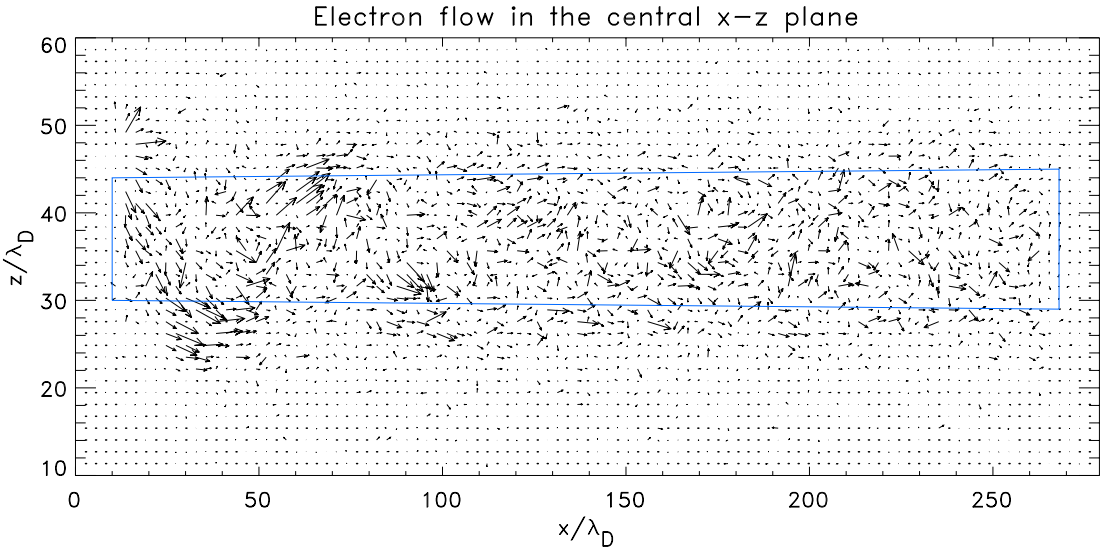

Figure 4.46: The electron mass flow in the central x-z plane for $\eta = 2$.

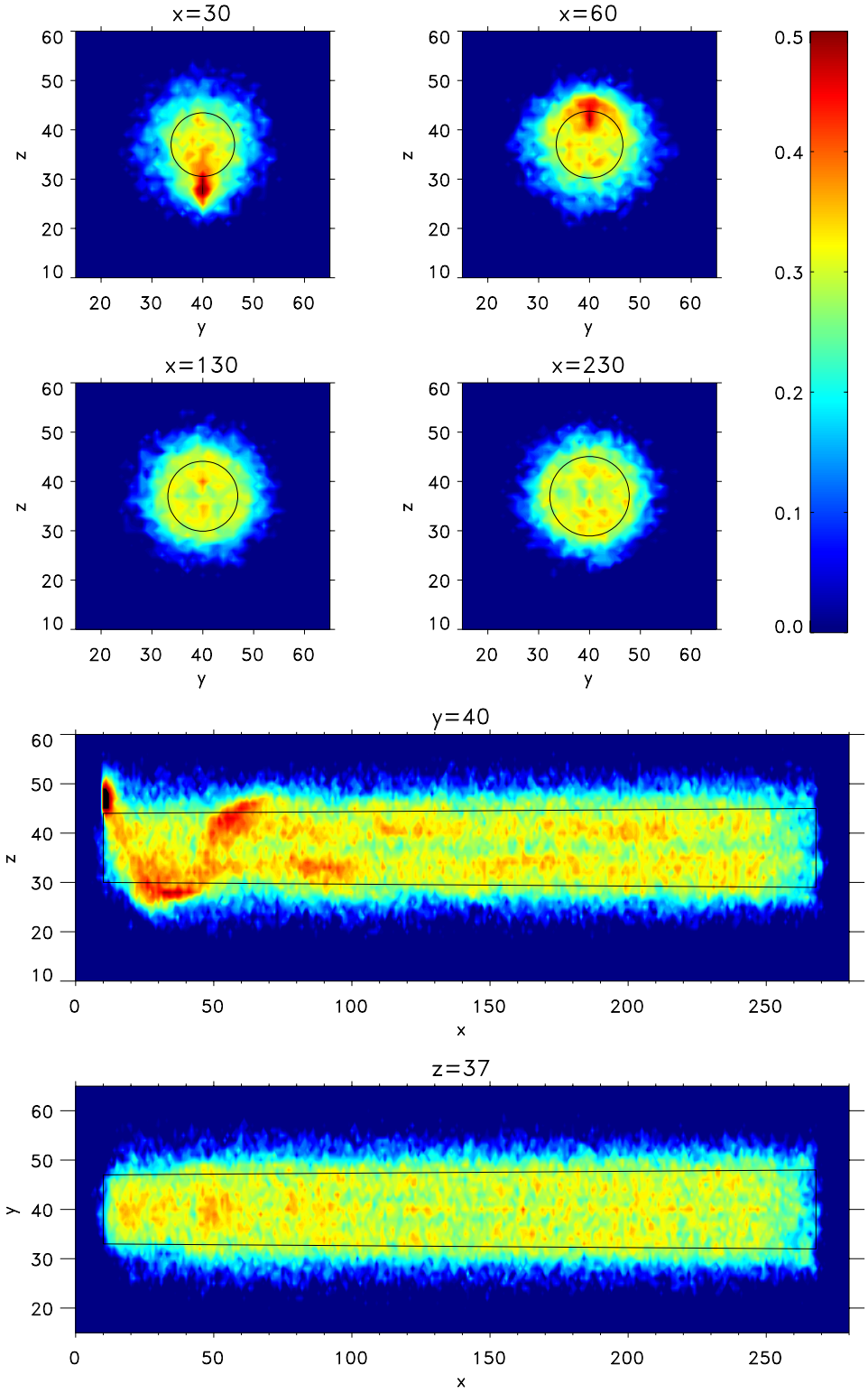

Figure 4.47: Electron density cuts for $\eta = 2$.

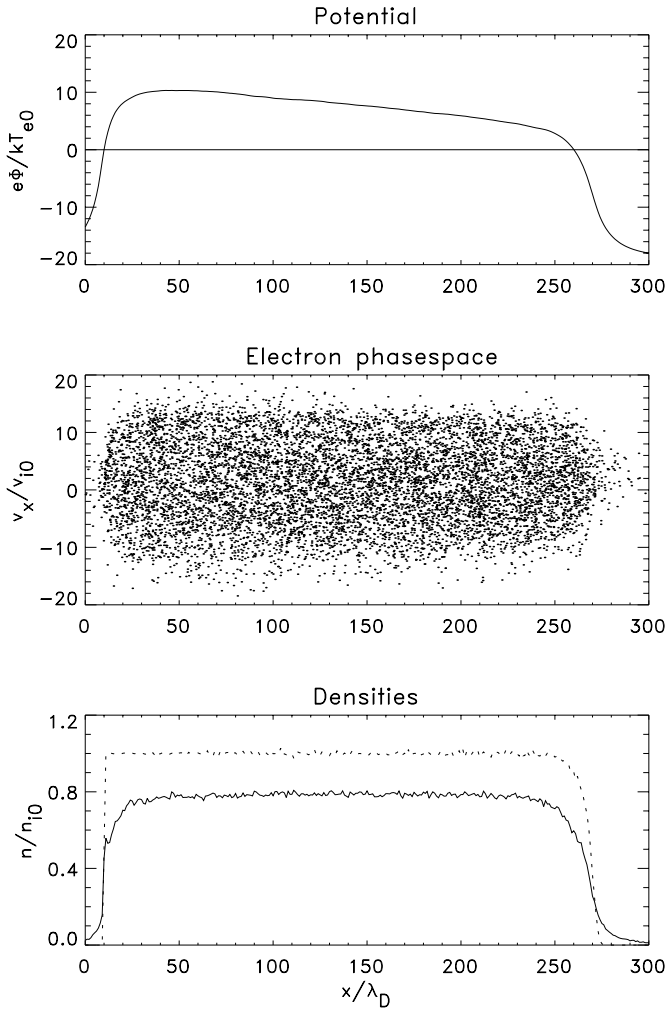

Figure 4.48: Same as Fig. 4.37 but with $\eta = 4$.

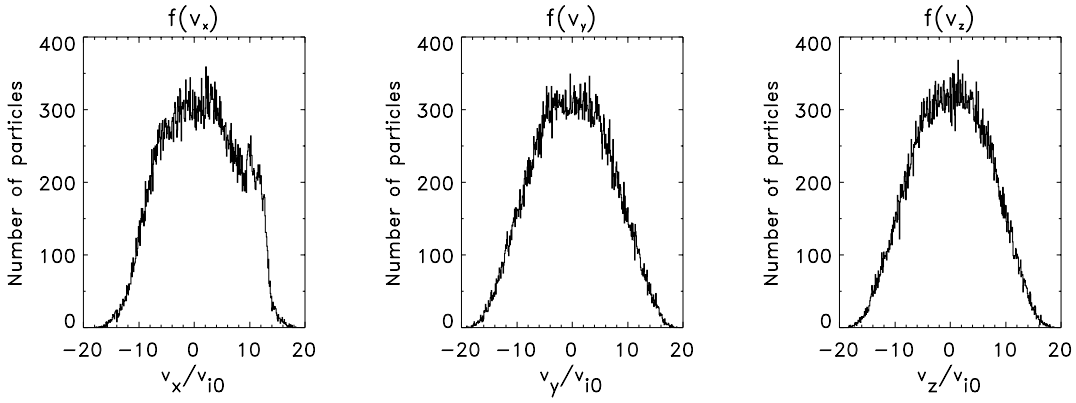

Figure 4.49: The three components of the electron velocity distribution function at $x = 220\lambda_D$ for $\eta = 4$.

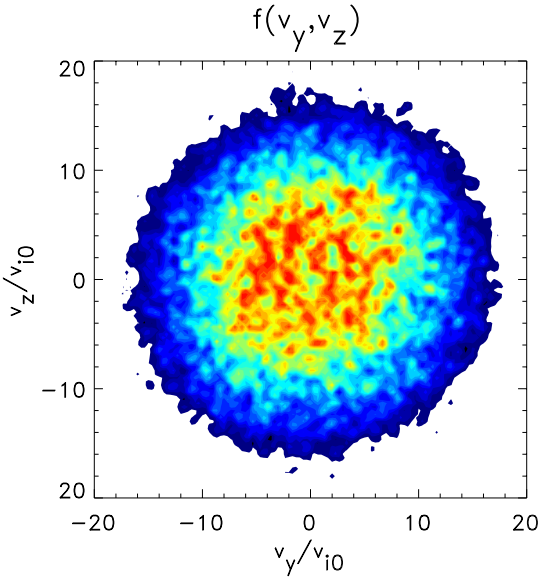

Figure 4.50: The electron distribution function of the perpendicular velocity components v_y and v_z in the region between $x = 180\lambda_D$ and $x = 220\lambda_D$ for $\eta = 4$.

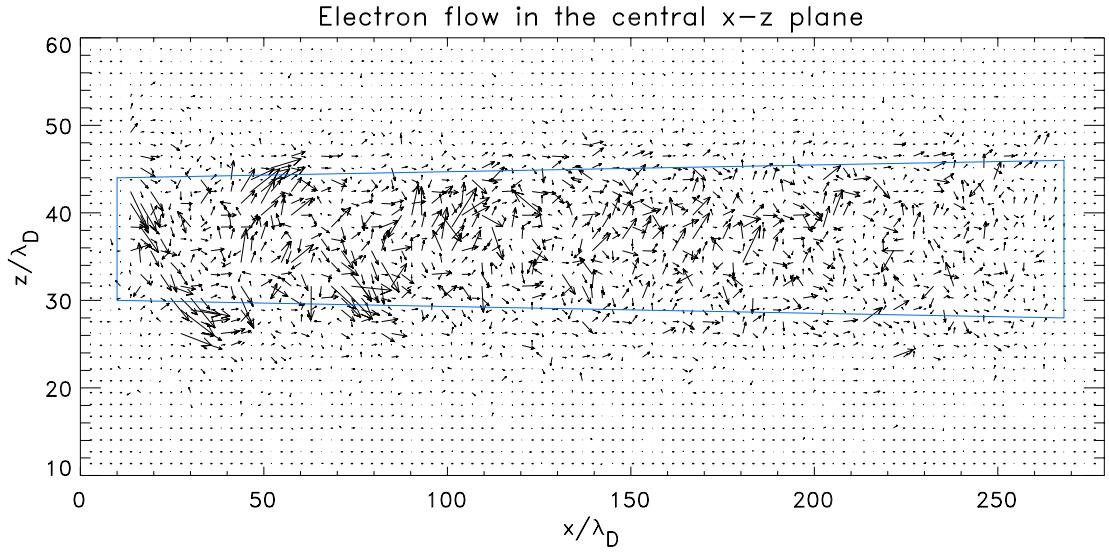

Figure 4.51: The electron mass flow in the central x-z plane for $\eta = 4$.

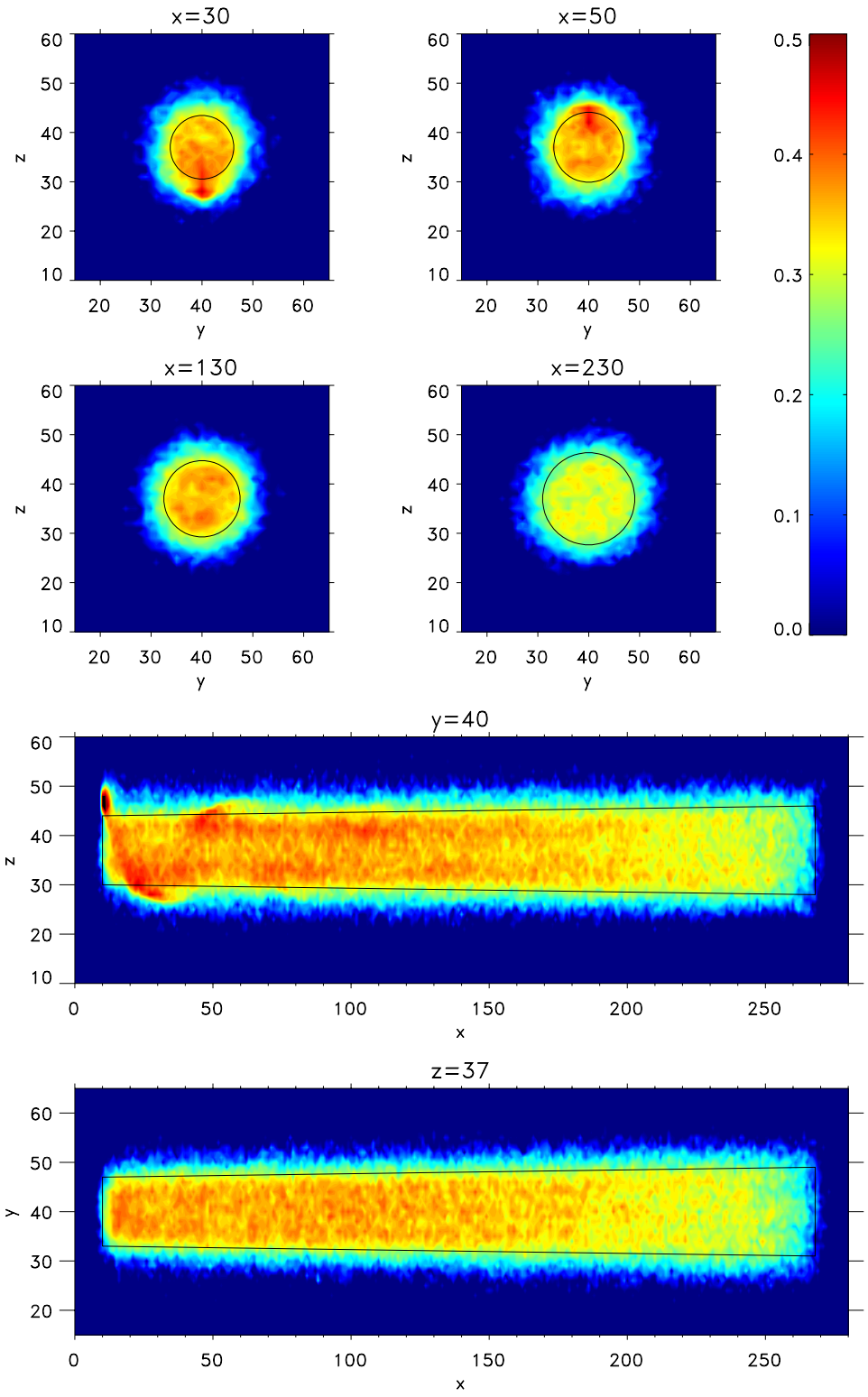

Figure 4.52: Electron density cuts for $\eta = 4$.

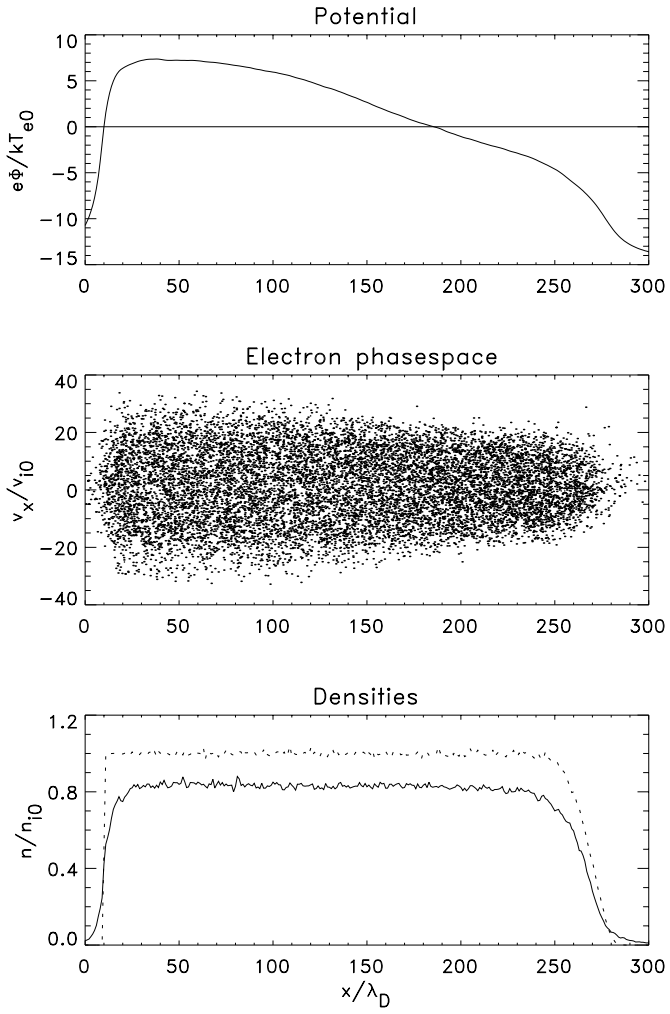

Figure 4.53: Same as Fig. 4.37 but with $\eta = 8$.

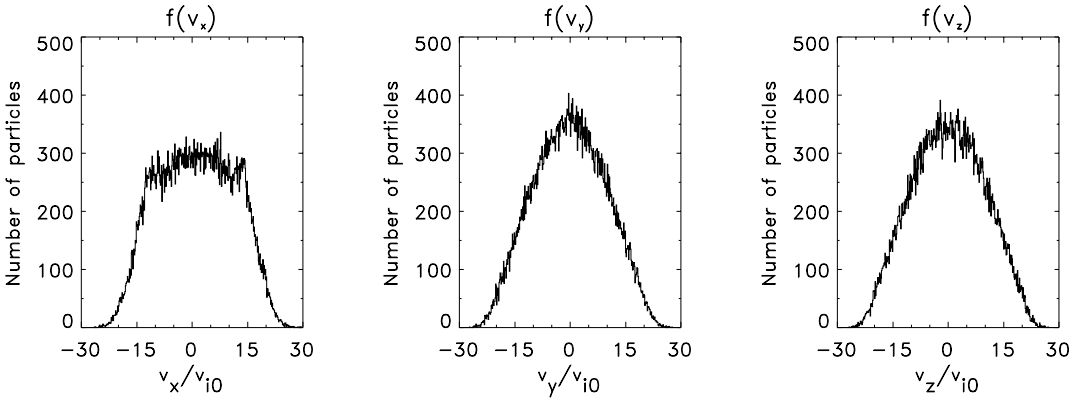

Figure 4.54: The three components of the electron velocity distribution function at $x = 220\lambda_D$ for $\eta = 8$.

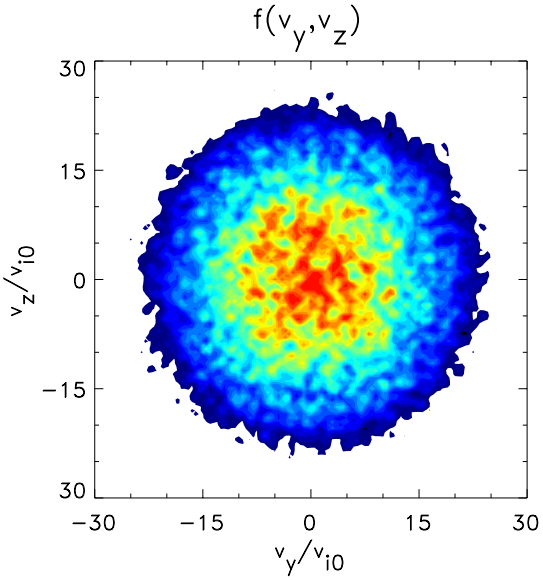

Figure 4.55: The electron distribution function of the perpendicular velocity components v_y and v_z in the region between $x = 180\lambda_D$ and $x = 220\lambda_D$ for $\eta = 8$.

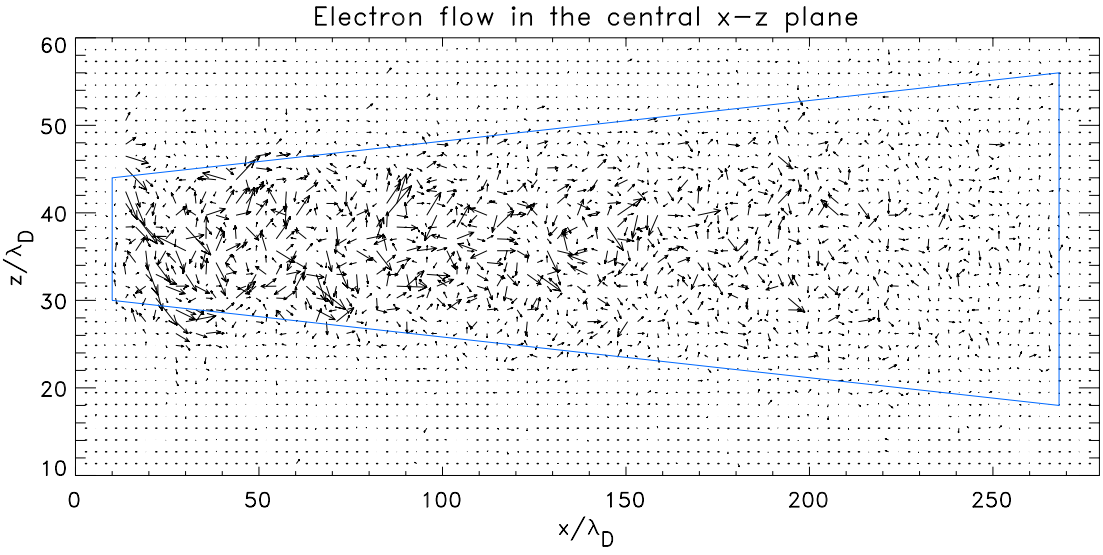

Figure 4.56: The electron mass flow in the central x-z plane for $\eta = 8$.

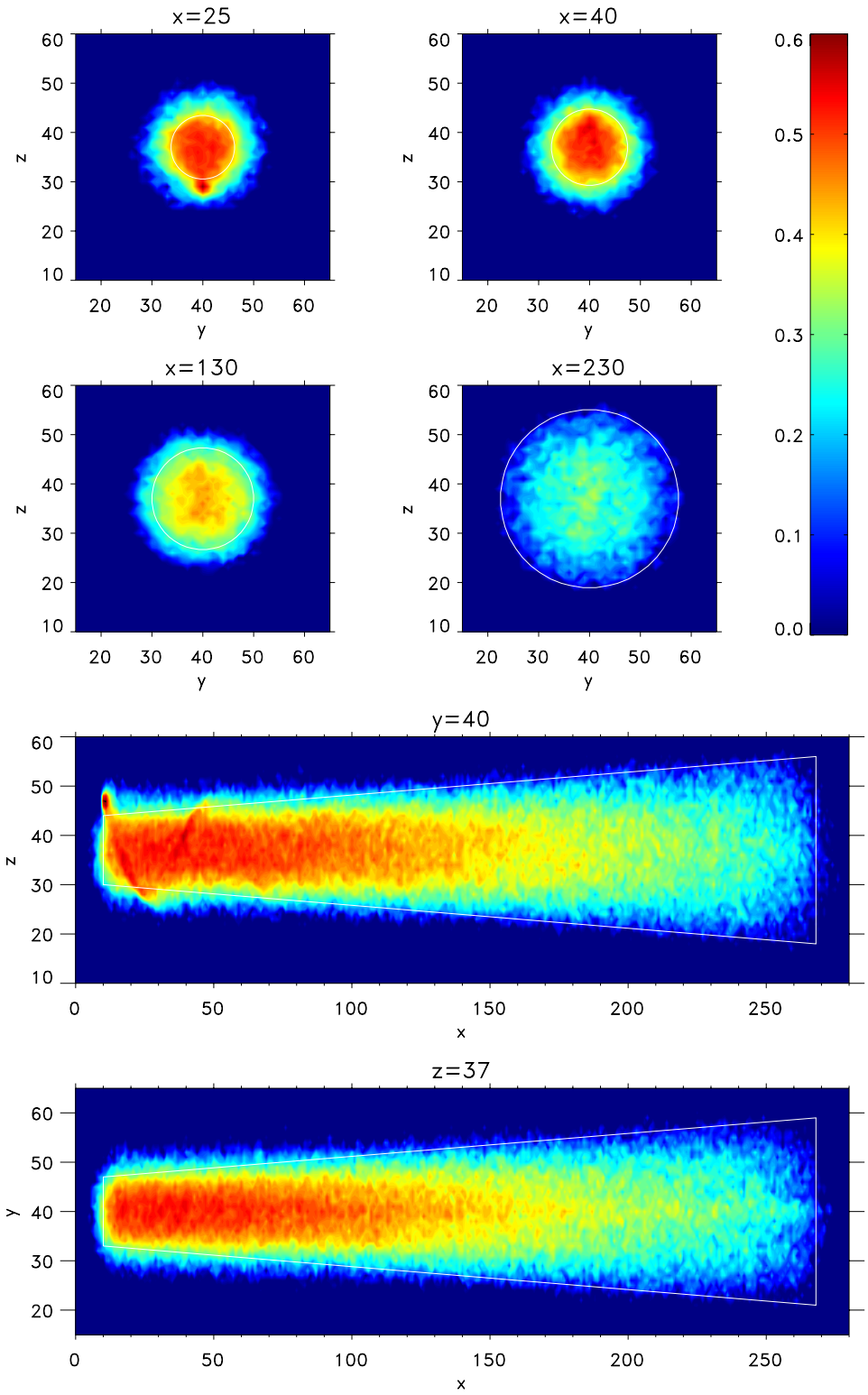

Figure 4.57: Electron density cuts for $\eta = 8$.

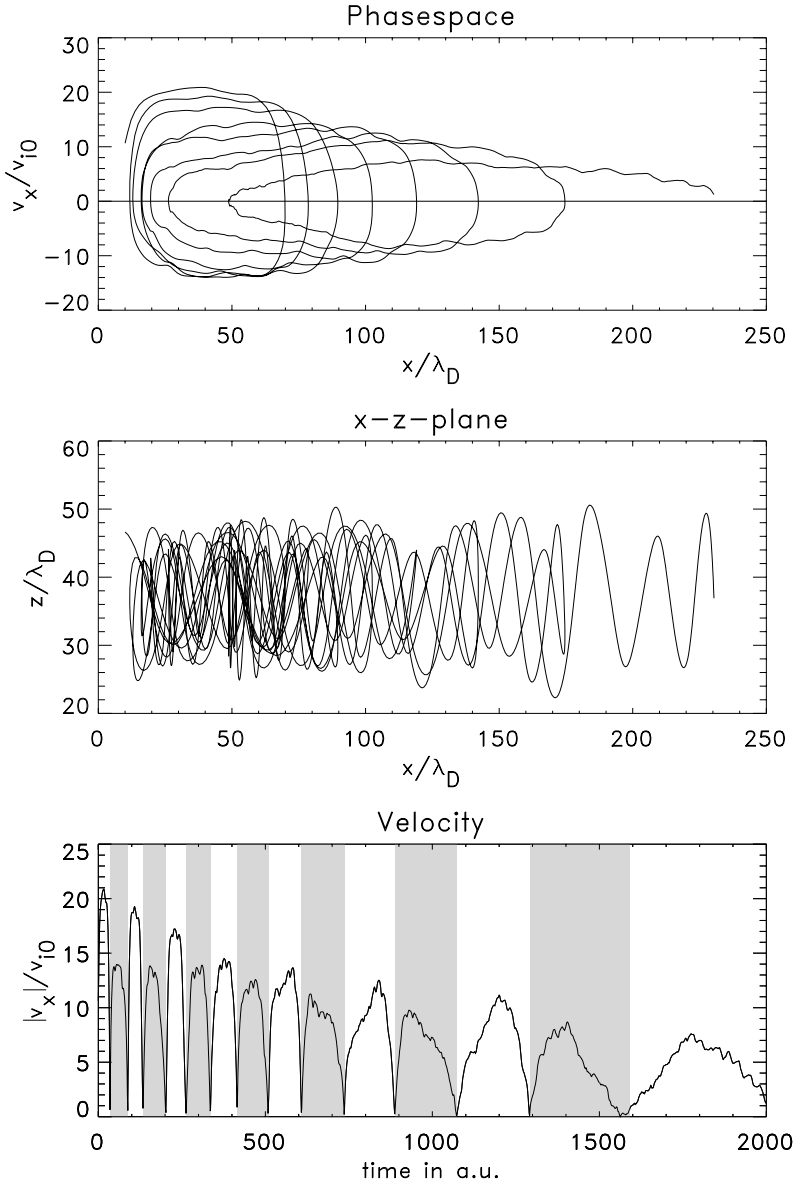

Figure 4.58: Dynamics of a representative test electron for $\eta = 8$. The fluctuations along the phase space path $v_x - x$ on a length scale of about $15\lambda_D$ (top panel) are due to electron oscillations in the lateral directions (middle panel). The white and grey sections in the third panel correspond to right-going and left-going legs of the electron path, respectively.

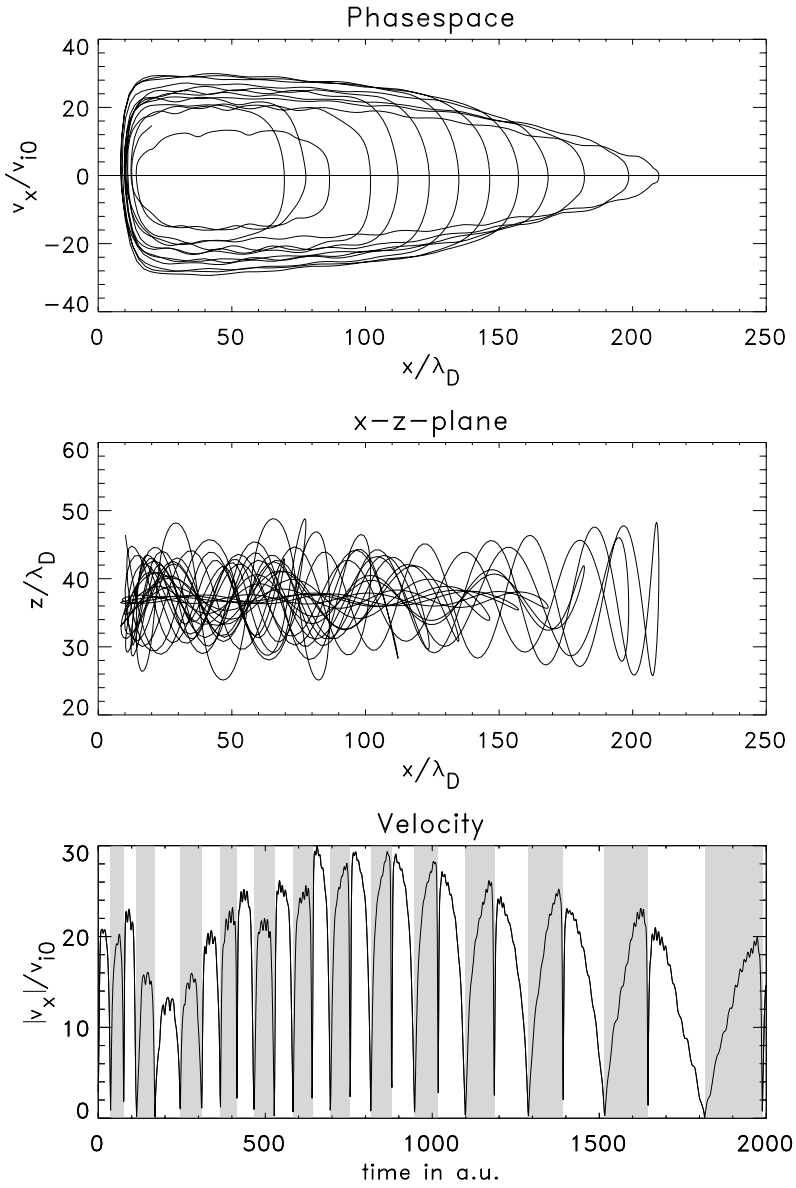

Figure 4.59: Same as Fig. 4.58, but of another, non-representative electron, which transitionally gains kinetic energy in the axial direction. The white and grey sections in the third panel correspond again to right-going and left-going legs of the electron path.

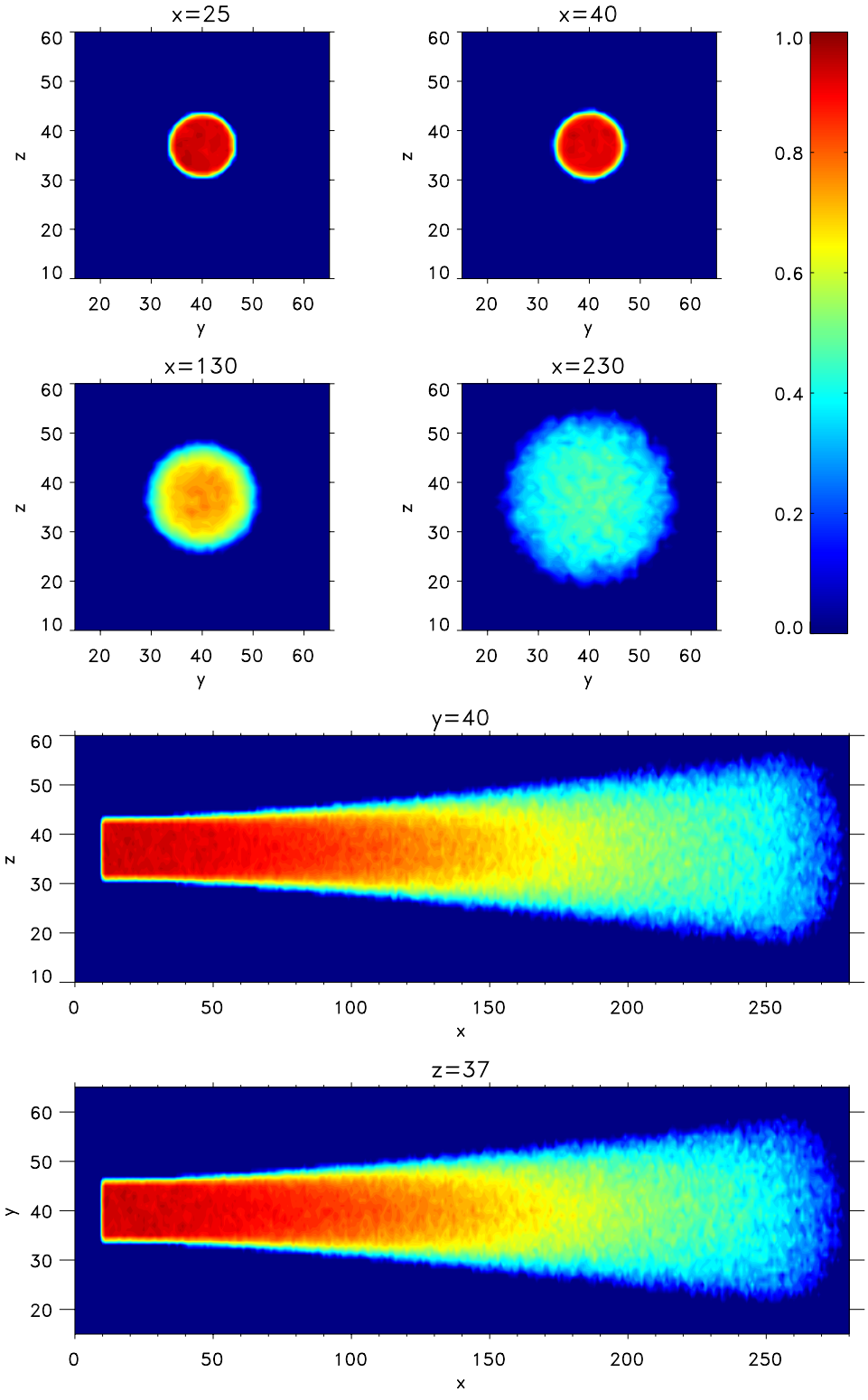

Figure 4.60: Ion density cuts for $\eta = 8$. The density values are normalized to n_{i0}. Note the lateral expansion of the beam.

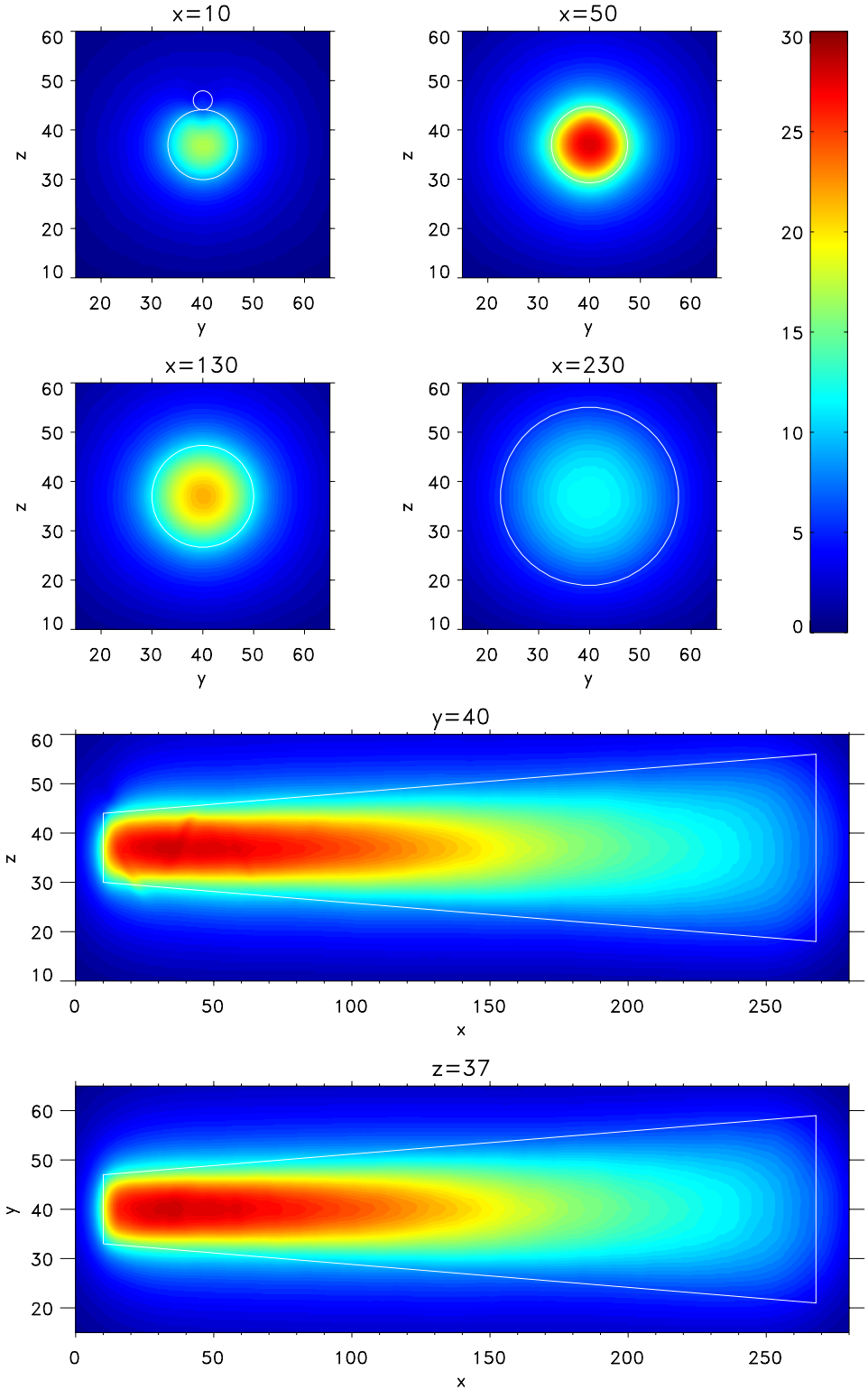

Figure 4.61: The potential for $\eta = 8$ with respect to the ambient level. The potential values are given in units of $\Phi_0 = k_B T_{e0}/e$.

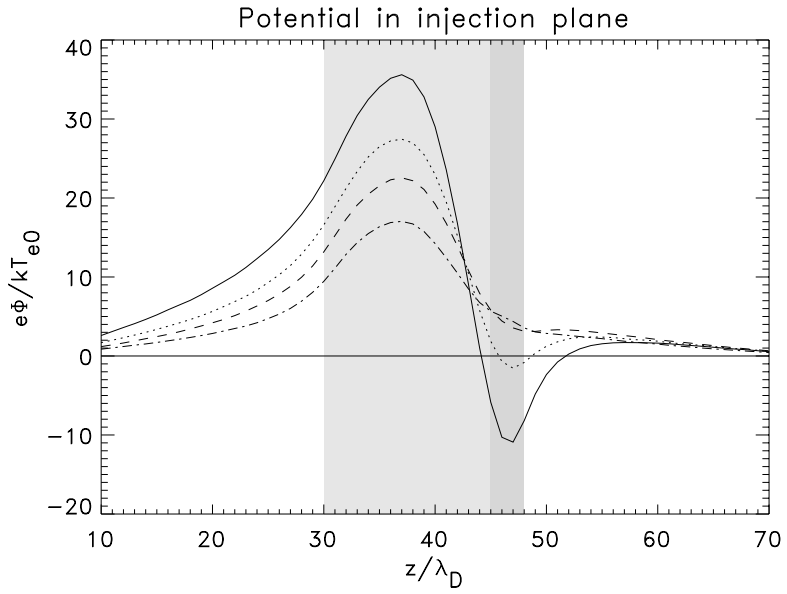

Figure 4.62: The potential in the injection plane at $y = 40$ for $\eta = 1$ (solid line), 2 (dotted), 4 (dashed) and 8 (dash-dotted). The ambient potential level is defined to be $\Phi = 0$. The ion and electron injection areas are marked in light and dark grey, respectively.

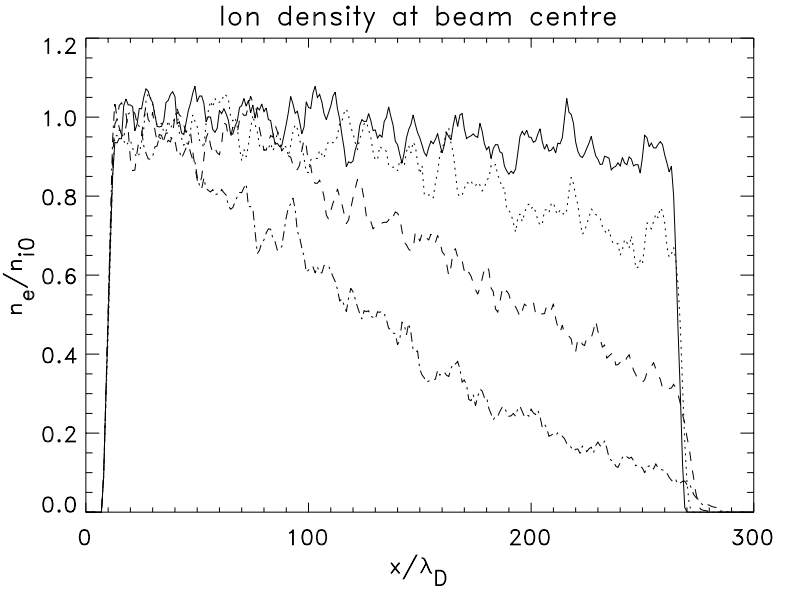

Figure 4.63: The ion densities in the centre of the beam for $\eta = 1$ (solid line), 2 (dotted), 4 (dashed) and 8 (dash-dotted). They are normalized to the nominal ion density of a non-diverging beam n_{i0}. Note the strong axial decrease for higher η, which is associated with lateral beam expansion and axial velocity dispersion.

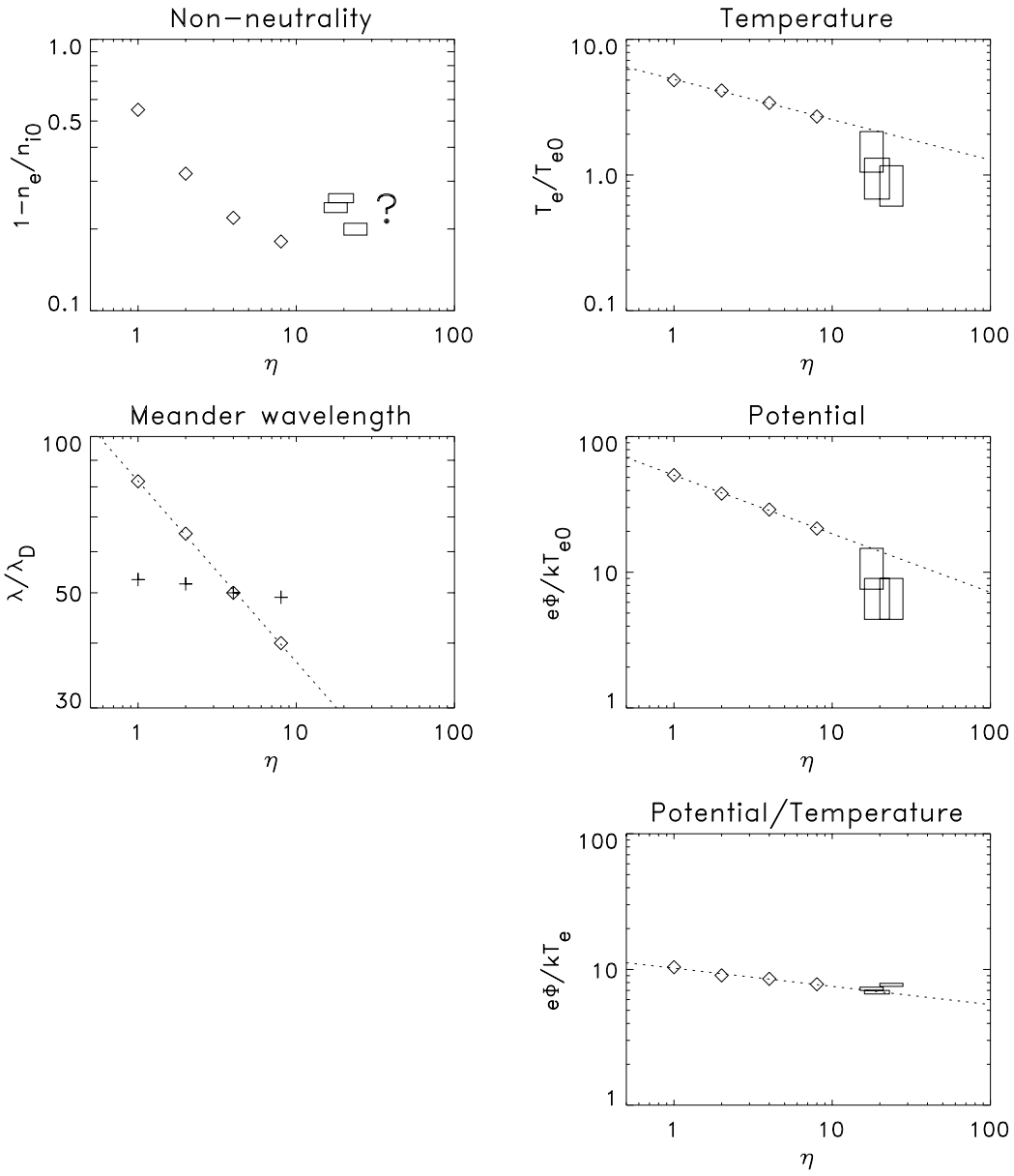

Figure 4.64: The η-dependence of some fundamental beam quantities (double-logarithmic plots): Simulation results from runs 1-4 ($\diamond$), meander wavelengths according to Eqn. (4.42) ($+$), and derived values from DS1 data (boxes, see Sec. 4.3.5). The dotted lines represent the relations (4.43)-(4.45) as obtained from a logarithmic regression of the simulation results. The box sizes for the experimental values do not reflect the total error, but only the uncertainty in T_{e0}, which cancels out in the ratio Φ/T_e. Note the only slight variation of this ratio over the whole η-range, as compared to T_e and Φ themselves. The experimentally determined non-neutralities (first panel) are based on the magnetometer readings and are of questionable value (see Sec. 4.3.5 for a detailed discussion).

The wavelength λ of the meandering movement decreases for increasing η according to

$$\lambda/\lambda_D = 81 \cdot \eta^{-0.35} \,. \tag{4.45}$$

As can be seen from Fig. 4.64, Eqn. (4.42) still gives the right order of magnitude for λ, but it does not reflect very well the decreasing trend for increasing η.

The weak point of Eqn. (4.42) is to choose the right streaming velocity of the newly injected electrons $\hat{v}$. For calculating the meander wavelengths from Eqn. (4.42), we simply picked $\hat{v}$ around the maximum velocity in the $v_x - x$ phase space, assuming that this velocity belongs to the newly injected, meandering particles. While this assumption is clearly justified in the case of run 1 (middle panel of Fig. 4.37), the phase space trace of the test electron of Fig. 4.59 showed that some electrons might undergo a later acceleration. Hence, the maximum velocity is not necessarily that of the newly injected electrons and their streaming velocity $\hat{v}$ to enter Eqn. (4.42) has to be chosen somewhat smaller.

While in runs 2 and 3, the electrons still tend to avoid the ion beam centre – albeit not as extremely as in run 1, the electron density in run 4 actually peaks in the centre and decreases radially outward. In the discussion of run 1 in the previous section, we explained the observation that the beam centre is practically void of electrons with the high kinetic energy of the electrons: It makes them circulate around the beam rather than reside in its centre. Since the kinetic energy gained when falling into the potential trough of the ion beam is decreasing for increasing η, it does not surprise that the electrons' tendency to avoid the beam centre gradually vanishes, in accordance with the disappearance of the ring character of the perpendicular velocity distribution (see above).

The longer simulation times of runs 3 and 4 allow to observe an appreciable lateral expansion of the ion beam. From Fig. 4.60, the expansion velocity $v_\perp^b$ in run 4 can be estimated as

$$v_\perp^b \approx \frac{R(x=200) - R(x=10)}{200-10} \cdot v_{i0} \approx \frac{17 - 7.5}{190} \cdot v_{i0} \approx 0.05 v_{i0} \,, \tag{4.46}$$

where $R(x)$ is the beam radius as determined from Fig. 4.60.

The radial expansion of a *quasi-neutral* plasma beam is driven by the thermal electron pressure. According to Wang and Hastings [1992] and Wang et al. [1996], the velocity of the expansion is roughly equal to the ion acoustic velocity c_s. Since

$$c_s = \sqrt{\frac{kT_e}{m_i}} = \sqrt{\frac{m_e}{2m_i}} \cdot \sqrt{\frac{2kT_e}{m_e}} = 1.4 \cdot 10^{-3}\, v_{e0}^{th} \tag{4.47}$$

and $v_{e0}^{th} = 8 v_{i0}$ for run 4, the ion acoustic velocity of this simulation is

$$c_s \approx 0.01 v_{i0} \,. \tag{4.48}$$

This is by a factor of 5 smaller than the observed expansion velocity $v_\perp^b$. Hence, the thermal electron pressure alone cannot account for the actual beam expansion. A second and more important driving force is the mutual electrostatic repulsion of the ions as a consequence of the *non-neutrality* of the beam. In order to

compute the expansion velocity associated with this repulsion, we consider again an infinitely long, charged beam: The electric field of a cylindrically symmetric space charge distribution is

$$E(r) = \frac{\tilde{Q}}{2\pi\varepsilon_0 r}\,, \tag{4.49}$$

where $\tilde{Q}$ is the charge per unit length inside the circle of radius r. For an electric charge sitting at the border of the beam, $\tilde{Q}$ is equal to the total beam charge per unit length, and r is the – momentary – beam radius R. The total charge per unit length $\tilde{Q}$ can be calculated from the charge density and the radius of the initial, non-expanded beam:

$$\tilde{Q} = \pi R_0^2 \rho_0\,. \tag{4.50}$$

Hence, the electric field as seen by a particle travelling always at the border of the expanding cylinder $r = R$ is

$$E(R) = \frac{\rho_0}{2\varepsilon_0} \cdot \frac{R_0^2}{R}\,, \tag{4.51}$$

leading to a potential $\phi(R)$ of

$$\phi(R) = -\int_{R_0}^{R} E(R')dR' = -\frac{\rho_0 R_0^2}{2\varepsilon_0} \cdot \ln\frac{R}{R_0}\,. \tag{4.52}$$

The velocity gained by the outermost ions upon expansion of the beam can thus be calculated as

$$v_i(R) = \sqrt{\frac{e\rho_0 R_0^2}{m_i\varepsilon_0} \cdot \ln\frac{R}{R_0}}\,. \tag{4.53}$$

It depends on the momentary radius of the beam and increases monotonously. This velocity v_i is to be identified with the observable expansion velocity of the ion beam $v_\perp^b$. By substituting ρ_0 with the ion plasma frequency and other appropriate relations, Eqn. (4.53) can be transformed into

$$v_\perp^b(R) = \sqrt{\kappa \cdot \frac{m_e}{2m_i} \cdot \ln\frac{R}{R_0}} \cdot \frac{R_0}{\lambda_D} \cdot \eta \cdot v_{i0}\,. \tag{4.54}$$

Here, κ is the degree of non-neutrality *inside* the beam:

$$\kappa = 1 - \frac{n_e}{n_i}\,, \tag{4.55}$$

with n_e and n_i averaged over the whole beam cross section.

The observed value of $v_\perp^b$ was determined from run 4 as the average expansion velocity between $x = 10$ and $x = 200$. At $x = 200$, the beam radius has grown to $R = 17\Delta x = 17\lambda_D$. κ drops from around 0.6 behind the injection plane to 0.2 at $x = 200$. For an estimate of the average expansion velocity, we therefore assume $\kappa = 0.4$. With these values, Eqn. (4.54) yields an expansion velocity of approx.

$0.048v_{i0}$. This is already closer to the observed expansion velocity of $0.05v_{i0}$ than the ion acoustic velocity c_s.

Hence, the combined force of the electron thermal pressure and the Coulomb repulsion of the ions can explain the observed beam expansion quantitatively. With the still appreciable degree of non-neutrality *inside* the beam of $\kappa \geq 0.2$, the Coulomb repulsion accounts for the greater part of the total $v_\perp^b$.

4.3.4 Dependence on beam diameter

In our studies of quasi-1D configurations in Secs. 4.1 and 4.2, the lateral beam dimension turned out to have an impact on the neutralization process. The smaller surface-to-volume ratio of wider beams was found to lead to a higher neutralization degree and thus to a smaller shock velocity (Figs. 4.11, 4.12, and 4.22), while in the presence of a magnetic field, the ratio r_{ge}/b between the thermal electron gyroradius and the beam width controlled the downstream temperatures (Fig. 4.27).

In order to assess the impact of the beam diameter b in case of spatially separated particle sources, we intended to run a series of simulations with an electron orifice diameter fixed at $3\lambda_D$ and varying ion beam diameters. However, for diameters of more than $20 - 25\lambda_D$, we encountered some problems with our simulation: In contrast to the quasi-1D simulations, where the electron density was rather confined to the ion beam and its immediate vicinity, the electrons in the 3D configuration of Fig. 4.36 tend to spread out very far from the beam centre (e. g. Fig. 4.41). A significant part of them does actually not escape but returns to the beam and contributes to its neutralization. Therefore, credible simulation runs require to accommodate these extensive particle orbits within the simulation box and thus to choose the box dimensions large enough. As in our 3D configuration the electron source moves farther out for wider beams, the problem becomes increasingly critical for growing beam diameters, and the required box dimensions do not allow a sensible simulation even on 128 PEs.

Hence, the role of the beam diameter in the case of separated particle sources remains an open question. We note, however, that for high η, the electron distribution within the beam is already quite homogeneous and does not reflect anymore its origin from a small source located outside the ion beam (Fig. 4.57). Apart from the single meander adjacent to the injection plane, it rather resembles the density distribution of a simple 1D injection geometry. For that reason, one might assume that in case of high η, the quasi-1D simulations can approximate the configuration with separated particle sources, at least as far as the *variation* of fundamental beam quantities with the beam diameter is concerned.

We therefore went back to our quasi-1D geometry of Fig. 4.1 and carried out a series of simulations with η fixed at 8 and varying beam widths of $b = 7.5, 15, 30,$ and $60\lambda_D$, which correspond to 0.5, 1, 2, and $4l_e$, respectively. The results for the

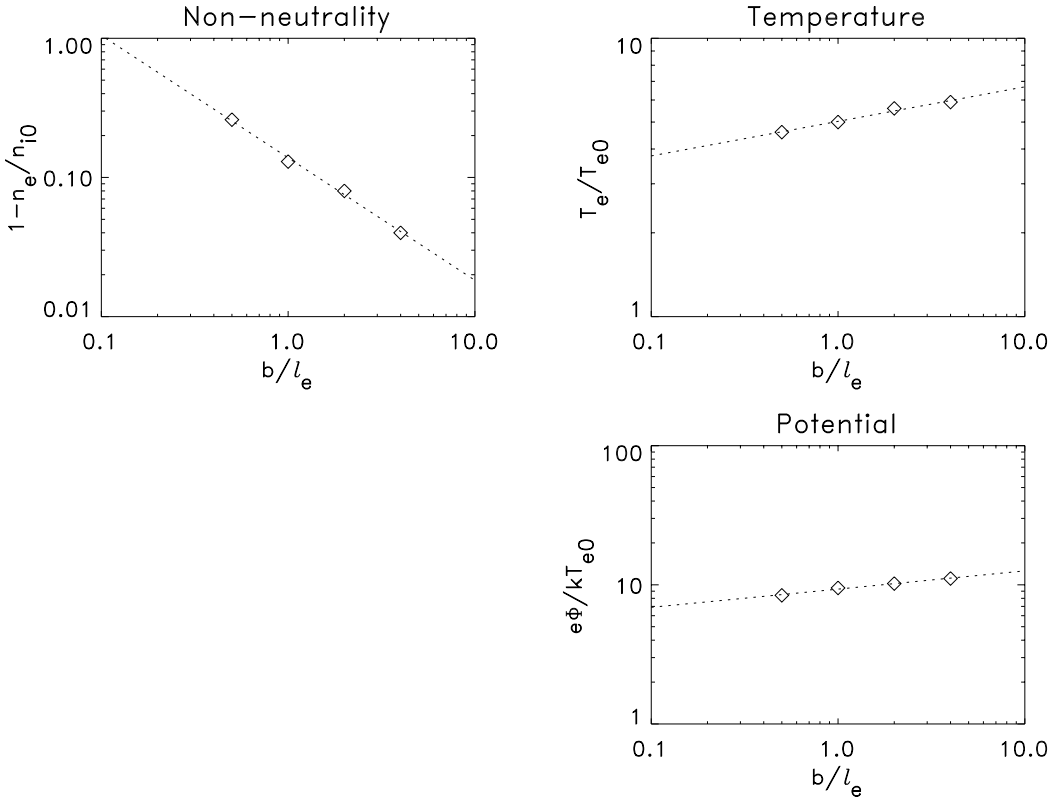

Figure 4.65: Degree of non-neutrality, electron temperature and potential with respect to the ambient for quasi-1D simulations according to Fig. 4.1 ($\eta = 8$ for all runs, double-logarithmic plots).

degree of non-neutrality, the electron temperature and the potential with respect to the ambient are displayed in Fig. 4.65.

A logarithmic regression of the data shown in Fig. 4.65 yields the following relations:

$$1 - n_e/n_{i0} = 0.14 \cdot (b/l_e)^{-0.88} \quad (4.56)$$

$$T_e/T_{e0} = 5.0 \cdot (b/l_e)^{0.12} \quad (4.57)$$

$$\Phi/\Phi_0 = 9.3 \cdot (b/l_e)^{0.12}\,. \quad (4.58)$$

Hence, with a power of 0.12, temperature and potential are relatively independent of the beam width, whereas the impact of b on the degree of non-neutrality is significant: $1 - n_e/n_{i0}$ decreases almost proportional to $(b/l_e)^{-1}$.

In view of the later comparison of our simulation results with actual measurements of DS1, we have cast Eqns. (4.56)-(4.58) in b/l_e rather than in b/λ_D. The Debye length λ_D is a numerically very important parameter (Sec. 2.7) and certainly plays a physical role in our simulations. It is, however, a quantity of a quasi- or nearly *neutral* plasma and, as such, starts to become physically significant once

electrons and ions are already mixed. Therefore, at least for the simulations with spatially separated particle sources, where an initially *non-neutral* plasma is injected, the electron inertia length l_e can be expected to be the more relevant length scale. That l_e plays indeed a major physical role can be deduced from the fact that typical scales in our simulations such as the width of the shock (Sec. 4.1) or the thickness of the injection sheaths (Secs. 4.1 and 4.2) are in the order of l_e ($14\lambda_D$ in our simulations).

Hence, it seems more sensible to base the rescaling of simulated quantities to "real values" on the ratio of the respective electron inertia lengths rather than on the Debye length ratio. In the case of the DS1 thruster, which has a diameter of $b = 0.3\text{m} = 3.75 l_e$, this ratio amounts to $3.75/1.1 = 3.4$ for simulation runs with a beam diameter of $b = 15\lambda_D = 1.1 l_e$, which is the most commonly used in this work. For $\eta \geq 8$, Eqns. (4.56)-(4.57) then allow to obtain an estimate of the respective rescaling factors for non-neutrality, temperature and potential. They are 0.31, 1.2 and 1.2, respectively, for the rescaling from our simulation to DS1, and are of interest for the comparison between the simulation results and the DS1 measurements in the next section.

4.3.5 Comparison with DS1 data

The relations (4.43)-(4.45) that were obtained from the series of simulation runs of Sec. 4.3.3 reflect the impact of the velocity ratio η on the various beam quantities. Although not being as decisive as in the case of a quasi-1D geometry (Secs. 4.1 and 4.2), η still seems to be an important parameter for the overall beam behaviour also in the case of spatially separated particle sources. In general, a higher η leads to a lower degree of non-neutrality, colder electrons and a lower potential with respect to the ambient. The electron meanders become shorter and less distinct. While T_e, Φ, and λ follow the obtained power laws very well (Fig. 4.64), the electron density exhibits a more complicated behaviour and only slowly approaches a complete neutralization.

Eqns. (4.43)-(4.45) can be used to extrapolate the plasma parameters towards higher values of η and thus allow a comparison of the simulation results with actual measurements aboard DS1. A quite complete set of DS1 data was recorded during the so-called "S-Peak": an operation specifically designed for ion engine diagnostics, where the thruster steps consecutively through three different thrust levels [Wang et al., 2000].

The S-Peak operation was started on January 22, 1999, at 21:36 UT and lasted until 22:17 UT of the same day. Among the instruments that collected data during the S-Peak are a Langmuir probe (LP), a retarding potential analyzer (RPA) and the pair of inboard and outboard fluxgate magnetometers (IB, OB). LP and RPA are co-located at about 0.5 m from the thruster centre (Fig. 4.66), while IB and OB are situated on a boom at a radial distance of 0.8 m and 0.9 m from the thruster centre, respectively.

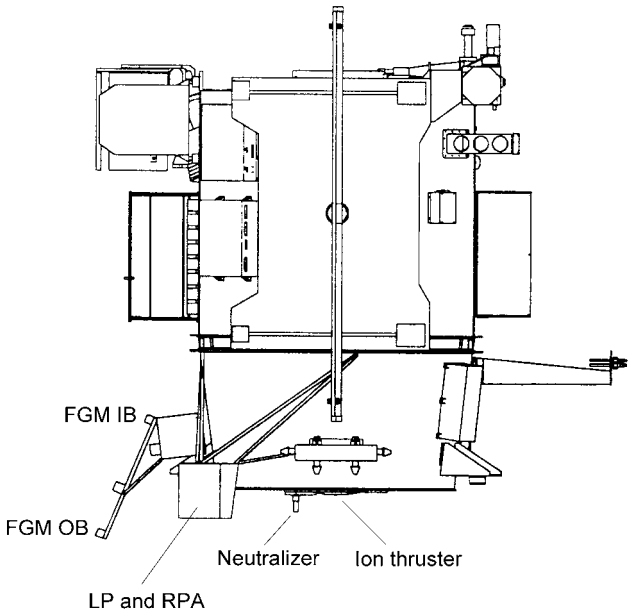

Figure 4.66: Side-view of the DS1 spacecraft. Note the locations of the fluxgate magnetometers (FGM IB and OB), Langmuir probe (LP) and retarding potential analyzer (RPA).

LP measures the electron temperature T_e as well as the difference between local potential Φ_{loc} and spacecraft ground Φ_{sc}. The potential of the thruster beam Φ_{beam} relative to spacecraft ground is determined by RPA from measurements of the kinetic energy of incoming CEX ions: As these are born cold within the ion beam, their kinetic energy upon reaching the grounded RPA is directly related to $\Phi_{beam} - \Phi_{sc}$. Hence, combining RPA and LP data, one can calculate the difference between the local and the beam potential [Wang et al., 2000]:

$$\Delta\Phi = \Phi_{beam} - \Phi_{loc} = (\Phi_{beam} - \Phi_{sc}) - (\Phi_{loc} - \Phi_{sc})\,. \tag{4.59}$$

$\Delta\Phi$ serves as a proxy for the beam potential with respect to the ambient. It probably underestimates the actual Φ_{beam}, because at the location of LP, the potential has probably not yet decayed to the ambient level (i. e. $\Phi_{loc} > 0$). Also the electron temperature measurements are likely to underestimate the actual T_e: Since LP is located outside the beam, the electrons reaching LP have already lost a part of their kinetic energy upon "climbing" out of the potential trough.

Judging from our simulations, the electrons can be expected to have the same average velocity as the ions. Therefore, any non-neutrality of the thruster beam constitutes an electric current, which should show up in the OB magnetometer readings (Fig. 4.66). As the magnetic field signature of such a current is largest in the azimuthal direction, we based our estimate of the non-neutrality $\kappa = 1 - n_e/n_{i0}$ on the azimuthal component of OB. It has to be noted that this method

is of questionable value, especially in view of the magnetically very disturbed environment of DS1 [Richter, 2000; Richter et al., 2001].

A more credible way of estimating the degree of non-neutrality is to assume the thruster beam again as an infinitely long, charged cylinder and to use Eqn. (4.33) for the electric field of such a cylinder in order to relate the measured potential difference to the charge density of the beam:

$$\Delta\Phi = \frac{\rho R^2}{4\varepsilon_0} \cdot \left(1 + 2\ln\frac{r}{R}\right) . \tag{4.60}$$

Here r, R, and ρ are the radial position of the Langmuir probe, the thruster beam radius and the charge density within the beam, respectively.

Table 2 summarizes the measurements and the derived quantities during the three different thrust intervals of the S-Peak. The values of v_{i0}, n_{i0}, T_e, and $\Delta\Phi$ were provided by Wang et al. [2000], κ_{OB} is computed from the magnetometer records of Richter [2001], and $\kappa_{\Delta\Phi}$ is calculated from $\Delta\Phi$ as just outlined. The injection velocity ratio η is estimated by assuming an initial electron temperature T_{e0} of $1-2$ eV [suggested by Wang, 2001].

v_{i0} [km / s]	n_{i0} [$10^{15}\mathrm{m}^{-3}$]	η	T_e [eV]	$\Delta\Phi$ [V]	κ_{OB}	$\kappa_{\Delta\Phi}$
29.8	1.54	$20\ldots28$	1.17	9	0.2	$1.7\cdot10^{-5}$
37.0	1.96	$16\ldots23$	1.33	9	0.26	$1.3\cdot10^{-5}$
38.7	3.22	$15\ldots21$	2.09	15	0.24	$1.3\cdot10^{-5}$

Table 2: Measurements and derived beam quantities during the three thrust intervals of the DS1 S-Peak [partly after Wang et al., 2000, and Richter, 2001].

Before comparing them with the results obtained from our simulation, we emphasize that the beam quantities as derived from DS1 measurements are only of a very limited significance. They have been obtained in a rather indirect way, and the quality of the data from which they were derived is quite poor [Wang et al., 2000]. This is especially true for the magnetic field data [Richter, 2000 and 2001].

Therefore, the large discrepancy between κ_{OB} and $\kappa_{\Delta\Phi}$ does actually not surprise. As already noted, $\kappa_{\Delta\Phi}$ can be expected to be the more realistic value. According to Eqn. (4.54), a non-neutrality of the DS1 beam corresponding to $\kappa_{\mathrm{OB}} = 0.25$ would lead to lateral expansion velocities vastly exceeding v_{i0} directly behind the thruster exit, i. e. to a quasi-explosion of the beam, which is clearly not observed. When taking $\kappa_{\Delta\Phi} \approx 1.5\cdot10^{-5}$ as an estimate for the non-neutrality of the beam, one concludes from Eqn. (4.54) that roughly 1 m behind the DS1 thruster exit, the contribution of the mutual electrostatic repulsion to the lateral beam expansion

starts to dominate over that of the electron thermal pressure ($c_s \approx 0.03 v_{i0}$), which seems much more realistic.

The non-neutrality $\kappa_{\Delta\Phi}$ is much lower than we would expect from the simulation results (≈ 0.15, see Fig. 4.64), even after correction with the scaling factor of 0.31 that accounts for the difference between simulated and actual beam diameter (see previous section). Possible reasons for this disagreement are numerous: errors in the potential measurements, the crudeness of obtaining the value of κ from these measurements, or our scaling towards wider beams, which is based on a quasi-1D study with an η being much smaller than the one here. Hence, without more trustworthy measurements or numerical simulations of beam parameters closer to the DS1 values, it is impossible to draw conclusions on the actual degree of non-neutrality.

According to the quasi-1D study of the previous section, the impact of the beam diameter on temperature and potential is only minor. Having in mind that the measurements of electron temperature and potential are most likely underestimates of their actual values, one concludes from Fig. 4.64 that these quantities fit actually quite well into the respective relations (4.43) and (4.44). This becomes particularly clear when considering the *ratio* between potential and temperature Φ/T_e that partly compensates the underestimating of both quantities: It is almost perfectly consistent with our simulation results (Fig. 4.64, fifth panel). The fact that this ratio stays quite constant, while Φ and T_e themselves drop roughly by an order of magnitude for η ranging from 1 to around 20 (Fig. 4.64), is a consequence of their close connection that was noticed on several occasions already: Deeper potential troughs provide the electrons with more kinetic energy, i. e. with a higher T_e.

Hence, as far as the the data quality allows, the DS1 measurements confirm the injection velocity ratio η as being a major controlling parameter for the overall beam behaviour. What has to be investigated in more detail, however, is the role of the lateral beam dimension, which evidently has a strong impact on the degree of non-neutrality.

Chapter 5

Summary and outlook

This work was motivated by the recently revived interest in electric propulsion devices and had two objectives: (i) the development of a numerical simulation tool for investigating the process of ion thruster beam neutralization as well as the interaction of the neutralized beam with the solar wind, and (ii) the application of this tool for a detailed study of the plasma physics pertinent to the neutralization process.

The first objective was met by setting up the parallel 3D electromagnetic particle-in-cell simulation code ISOLDE. It solves the full set of Maxwell's equations and the particle equations of motion in a self-consistent manner. The code correctly describes kinetic processes, both electrostatic and electromagnetic, and is thus applicable to a wide variety of plasma physical scenarios in laboratory as well as in space plasmas.

A particularity of simulating ion thrusters with spatially separated electron and ion sources is the injection of an initially non-neutral plasma. To make such an injection consistent with a rigorously charge-conserving electromagnetic field solver was shown not to be a trivial task. Artificial divergences of the electric field emerge in case of inappropriate injection schemes. With our "generator", we have presented a technique that creates particles in accordance with charge conservation and thus allows a self-consistent injection of non-neutral plasmas.

Two MPI-based parallel versions of the code were developed. One employs a standard domain decomposition scheme and runs fairly efficiently on up to 8 processors. Based on this version, we obtained an optimized parallel code by incorporating a taskfarm for the dynamic distribution of the particle associated workload among the different processors. For the first time such a taskfarm-based load-balancing strategy was implemented into a plasma PIC simulation. It proved to be capable of dealing with the imbalance of particle work and yielded a parallel efficiency of about 0.8 for up to 128 processors.

An even better performance could be achieved by also out-sourcing the particle administration rather than only the particle associated computational work. In the present work, however, this idea was not pursued any further.

We employed our simulation code to carry out a systematic investigation of the plasma physical processes accompanying ion thruster beam neutralization. Starting off with a quasi-1D injection geometry, we continuously refined the simulation conditions by applying an external magnetic field, introducing a spatial separation between electron and ion source and considering a practically point-like electron emitter to finally arrive at a simulation scenario comparable to the actual ion thruster aboard Deep Space 1.

An important common feature of all these studies is the dominant role of the injection velocity ratio $\eta = v_{e0}^{th}/v_{i0}$ between electron thermal velocity and ion bulk velocity. Most remarkable is its impact in the case of quasi-1D geometries. Depending on the value of η, the overall beam behaviour switches between two completely different scenarios: For η being smaller than a critical value η_{crit}, a moving electrostatic shock of a few $k_B T_{e0}$ emerges, which generates a fully thermalized plasma, whereas for $\eta > \eta_{crit}$, no such shock occurs and the plasma does not get thermalized.

Based on a simple 1D model, we derived and successfully verified the following necessary condition for the occurrence of the shock:

$$v_{i0} > v_e^{th}\,, \tag{4.6}$$

where v_e^{th} is the downstream electron thermal velocity. The shock is excited via the same process that gives rise to a special family of electrostatic shocks, namely double layers: the reflection of electrons at a negative charge density spike or, in the case of the ion thruster, at the leading edge of the expanding ion beam.

According to our studies, an axial magnetic field does not have a significant impact on the quasi-1D neutralization process, unless it exceeds field strengths corresponding to $\Omega_e = 0.2\omega_{pe}^0$. But also for such strong fields, the shock-like neutralization scenario with a fully thermalized downstream plasma can be enforced by suitably choosing the injection velocity ratio η according to Eqn. (4.6). However, in contrast to the $B = 0$ case, the thermalization does not take place by virtue of a single electrostatic shock front, but rather via a non-linear, Landau-damped electrostatic wave. This essentially one-dimensional wave was identified as being excited by the difference between electron and ion bulk velocity upon injection.

When considering that the two major technical concerns of the operation of ion thrusters are (i) electromagnetic interference possibly induced via plasma instabilities and (ii) the thruster lifetime-limiting erosion of the grid system by backstreaming CEX ions, it becomes obvious why the shock-like neutralization regime can be of great technical importance: The shock-neutralization as revealed by our simulations generates a fully thermalized downstream plasma with no free energy to drive unwanted instabilities. Moreover, the potential drop associated with the shock front acts as a shield against backstreaming CEX ions and can thus effectively reduce the erosion of the thruster grid.

Today's ion thrusters are far away from fulfilling the necessary condition (4.6) for shock-like neutralization. They are operating at $\eta \gg 1$, whereas in the absence of

a magnetic field, η_{crit} turned out to be 1.7. The intentional application of a strong axial magnetic field was found not to yield a major relaxation of Eqn. (4.6) either. However, given the recent interest in electric propulsion, future thrusters might be capable of accessing the range around $\eta = 1.7$. Such thrusters would fulfill the necessary condition for shock-like neutralization and could then benefit from a very efficient neutralization.

In the shock-free regime, we investigated the effect of the spatial separation between electron and ion source. For this geometry, the potential profile resembles a rather featureless reversed potential trough, and the electrons do not adopt a Maxwellian distribution. After their injection, they perform a meandering movement between the top and bottom surface of the ion beam before spreading out more homogeneously. A sort of Fermi-deceleration between the expanding ends of the ion beam was found to be responsible for the adaptation of the average electron velocity to the ion bulk movement in the absence of collisions.

Albeit not as substantially as in the quasi-1D case, the injection velocity ratio η continues to control the electron behaviour also for spatially separated particle sources. In general, increasing η results in a higher degree of neutralization, lower beam potentials with respect to the ambient, colder electrons and a more homogeneous electron distribution within the ion beam. Based on our findings, we derived quantitative relations for the exact dependence of these parameters on η, which – in combination with the available measurements aboard Deep Space 1 – give a coherent picture of the impact of η on the overall beam behaviour.

Our work is the most thorough treatment of ion thruster beam neutralization to date. By studying this process from “first principles” up to more realistic configurations and paying special emphasis to the interdependence of the various plasma parameters, it does not only provide considerable insight into the plasma physics underlying neutralization, but also revealed a previously unknown neutralization regime: the shock-neutralization – a promising option for the design of future ion thrusters.

Due to numerical constraints, we had to restrict our simulations with spatially separated electron and ion sources to the shock-free regime. To confirm the possibility of shock-neutralization in this geometry remains an open task. Other aspects that deserve further attention are the role of the beam diameter in the case of spatially separated particle sources, the impact of particle collisions, and the beam behaviour on ion time scales. These issues, as well as an investigation of the possible interaction of the neutralized beam with the solar wind, will be left for future studies.

List of symbols

	Chapter 2
b	ion beam width
$\mathbf{B}$	magnetic field
$\mathbf{B}_0$	background magnetic field
$\mathbf{b}$	magnetic field perturbation
$\mathbf{b}_0$	$= q_s \mathbf{B}^n \Delta t / 2\gamma^n m_s$, normalized magnetic field
B^{rel}	relocated magnetic field
B_p	magnetic field interpolated to particle position
c	vacuum velocity of light
d	thickness of conductivity layer
e	elementary charge ($e > 0$)
$\mathbf{E}$	electric field
E_{el}	electric field energy
E_{kin}	kinetic energy
E_{mag}	magnetic field energy
E_{tot}	$= E_{kin} + E_{el} + E_{mag}$, total energy
E_p	electric field interpolated to particle position
E^{rel}	relocated electric field
$\mathbf{E}_{tang}$	electric field tangential to boundary
$\mathbf{F}$	force
i, j, k	grid indices for x, y, and z direction
$\mathbf{j}$	electric current
$\mathbf{j}_c$	conduction current
k	wave number
k_B	Boltzmann constant
l_e	electron inertia length
l_s	skin length
m_e	electron mass
m_i	ion mass
m_s	particle mass of species s
m_S	super-particle mass of species s
n_e	electron density
n_{i0}	ion density close to the injection plane (“nominal ion density”)

	Chapter 2 (contd.)
N	number of injected particles per time step
N_D	number of particles in a Debye sphere
N_S	number density of super-particles of species s
q	electric charge
q_s	electric charge of particle species s
q_S	electric charge of super-particle species s
Q	charge density of a particle (particle charge per particle volume)
Q_S	charge density of a super-particle
r_t	$= \Delta t_{max}/\Delta t$, temporal resolution
r_x	$= \Delta X/\lambda_D$, spatial resolution
S	particle shape factor
t	time
t_{inj}	individual particle injection time
Δt	time step
Δt_{max}	maximum time step according to Courant condition
T_e	electron temperature
T_{e0}	electron temperature upon injection
T_i	ion temperature
T_{tr}	Number of time steps for transfer (generator)
$\mathbf{u}$	$= \gamma \mathbf{v}$, relativistic particle velocity
$\mathbf{u}^-, \mathbf{u}^+$	$\mathbf{u}$ before and after rotation (particle push)
$\mathbf{v}$	particle velocity
v_{e0}^{th}	electron thermal velocity upon injection
v_{i0}	ion bulk velocity upon injection
$v_\perp$	perpendicular component of particle velocity (with respect to $\mathbf{B}$)
$\mathbf{v}_{tr}$	transfer velocity (generator)
ΔV	fraction of particle volume
w	weights for scatter / gather
$\Delta x, \Delta y, \Delta z$	particle displacements
$\Delta X, \Delta Y, \Delta Z$	grid spacings
X_i, Y_j, Z_k	coordinates of grid cell (i, j, k)
γ	relativistic correction: $\gamma = 1/\sqrt{1 - v^2/c^2}$
η	$= v_{e0}^{th}/v_{i0}$, injection velocity ratio
Θ	rotation angle of gyrating particle
λ_D	Debye length
λ_e	electron mean free path
λ_i	ion mean free path
ρ	charge density
ρ_p	charge density of point-like particles
ρ_S	charge density of super-particles
σ	electric conductivity
ν_{ee}	electron-electron collision frequency

	Chapter 2 (contd.)
ν_{ei}	electron-ion collision frequency
ν_{ii}	ion-ion collision frequency
ξ, η, ζ	normalized particle coordinates within a grid cell
χ	$= \Delta x \Delta y \Delta z$
ω_{lg}	frequency of longitudinal waves
ω_{tr}	frequency of transverse waves
ω_{pe}^0	electron plasma frequency for $n_e = n_{i0}$
ω_{ps}	plasma frequency of species s
ω_{pS}	super-particle plasma frequency of species s
ω_{pe}^S	super-particle plasma frequency of electrons
Ω_s	gyrofrequency of species s

	Chapter 3
`bndry(1:6)`	Boolean array containing whether or not a PE is situated next to a global boundary
`data`	the data to be sent
`finished`	Boolean variable, true if all the tasks are carried out
`found`	Boolean variable, true if master has found work for slave
`ihigh`, `jhigh`, `khigh`	upper domain boundaries
`ilow`, `jlow`, `klow`	lower domain boundaries
`message_type`	$= 1$ if slave reports his work, $\neq 1$ if slave reports the execution of a task
`myid`	a PE's id
`n_elements`	length of the data array to be sent or received
`nbour(1:6)`	array containing the ids of the neighbouring PEs
`nogx`, `nogy`, `nogz`	number of grid cells of global simulation grid
`nopr`	number of PEs used for a simulation run
`noprx`, `nopry`, `noprz`	number of PEs in each direction
`no_part(1:2)`	number of electrons and ions
`no_slabs`	number of slabs into which each PE domain is subdivided
`old_no_part(1:2)`	old number of electrons and ions
`part_n`	particle numbers
P	parallel efficiency
P_n^{fix}	P for a fixed problem size analysis on n PEs
P_n^{scal}	P for a scaled problem size analysis on n PEs
`rcv_buffer`	memory buffer to accommodate the incoming data
`sender_pe`	id of sending PE
`target_pe`	id of the target PE
`type`	MPI data type
T_n	run time on n PEs
T_{new}	run time of the taskfarm code
T_{old}	run time of the old code (straightforward parallelization)
`window_j`	handle for the current window
`window_part`	handle for the particle window
`window_rel_EB`	window handle for the relocated electromagnetic field
`work_package`	array specifying a task
`work_status`	array specifying the work status of all tasks
`xhigh`, `yhigh`, `zhigh`	particle coordinates of upper domain boundaries
`xlow`, `ylow`, `zlow`	particle coordinates of lower domain boundaries

	Chapter 4
b	ion beam width
c_s	ion acoustic velocity
d	electrode spacing (diode model)
D	"spring constant"
f	electron distribution function
J_0	$= en_0v_0$, electron current density (diode model)
J_c	Child-Langmuir critical current density (diode model)
k	wave number
k_B	Boltzmann constant
l_e	electron inertia length
n_{down}	downstream electron density
n_e	electron density
n_0	electron density (diode model)
n_{i0}	ion density close to the injection plane
Q	total electric charge
$\tilde{Q}$	electric charge per unit length
r	radial coordinate
r_{ge}	electron gyroradius
R	beam radius
R_0	initial beam radius
T_e	electron temperature
$T_{e\|\|}$	parallel electron temperature (with respect to $\mathbf{B}$)
$T_{e\perp}$	perpendicular electron temperature (with respect to $\mathbf{B}$)
T_{e0}	electron temperature upon injection
T_i	ion temperature
u	$= v_e/v_{i0}$, normalized velocity
$\hat{u}$	amplitude of u
u_0, u_1	$u = u_0 + u_1$
v_{av}	average velocity among forward streaming and reflected electrons
v_{down}	downstream electron velocity
v_{refl}	velocity of reflected electrons
v_e	electron velocity
v_e^{th}	electron thermal velocity
v_{th}	electron thermal velocity within shocklets
v_{e0}^{th}	electron thermal velocity upon injection
v_0	electron velocity at the electrodes (diode model)
v_i	ion velocity
v_{i0}	ion bulk velocity upon injection
v_{sh}	velocity of shock front
v_x	axial velocity of injected electrons
$\hat{v}$	$= v_x/v_{e0}^{th}$ normalized axial velocity
$v_\perp$	perpendicular component of particle velocity (with respect to $\mathbf{B}$)
$v_\perp^b$	lateral expansion velocity of the beam

	Chapter 4 (contd.)
v_φ	phase velocity
x_{vc}	position of virtual cathode
α	$= J_0/J_c$, current density ratio (diode model)
γ	damping decrement
δ	spatial displacement between electron and ion source
η	$= v_{e0}^{th}/v_{i0}$, injection velocity ratio
η_{crit}	critical injection velocity ratio
κ	$= 1 - n_e/n_i$, degree of non-neutrality
κ_{OB}	κ as determined from OB magnetometer measurements
$\kappa_{\Delta\Phi}$	κ as determined from potential measurements
λ_D	Debye length
ξ	$= v_{i0}/\omega_{pe}^0$, spatial scale
ρ	charge density
ρ_0	initial charge density (of the non-expanded beam)
τ	oscillation period of meandering movement
ϕ	electric potential on the border of a laterally expanding beam
φ	electric potential in the diode model
φ_0	$= mv_0^2/2e$, electrode potential (diode model)
Φ	electric potential
Φ_0	$= k_B T_{e0}/e$, normalized electric potential
Φ_{beam}	potential inside the ion beam
Φ_{loc}	local potential at Langmuir probe
Φ_{sc}	spacecraft ground potential
$\Delta\Phi$	$= \Phi_{beam} - \Phi_{loc}$
ψ	$= \Omega_e/\omega_{pe}^0$, frequency ratio
ω	oscillation period of meandering movement
ω_{pe}	electron plasma frequency
ω_{pe}^0	electron plasma frequency for $n_e = n_{i0}$
Ω_e	electron gyrofrequency

References

Amdahl, G. M., Validity of the single-processor approach to achieving large scale computing capabilities, AFIPS Conference Proceedings, **30**, AFIPS Press, Reston (Virginia), USA, p. 483, 1967.

Anderson, E., J. Brooks, Ch. Grassl, and S. Scott, Performance of the CRAY T3E Multiprocessor, CRAY Research Online Documents, http://www.cray.com/products/systems/crayt3e/paper1.html, 1996.

Anderson, E., J. Brooks, and T. Hewitt, The Benchmarker's guide to single-processor optimization for CRAY T3E systems, CRAY Research Publication, 1997.

Birdsall, C. K., Particle-in-cell charged particle simulations, plus Monte Carlo collisions with neutral atoms, PIC-MCC, *IEEE Trans. Plasma Sci.*, **19**, 65, 1991.

Birdsall, C. K., and W. B. Bridges, Electron dynamics of diode regions, Academic Press, New York, p. 68, 1966.

Birdsall, C. K. and A. B. Langdon, Plasma Physics via Computer Simulation, McGraw-Hill, New York, 1985.

Boris, J. P., Relativistic plasma simulation – optimization of a hybrid code, *Proc. Fourth Conf. Num. Sim. Plasmas*, Naval Res. Lab., Washington DC, p. 3, 1970.

Brinza, D., M. D. Henry, A. T. Mactutis, K. P. McCarty, J. D. Rademacher, T. R. van Zandt, J. J. Wang, B. T. Tsurutani, I. Katz, V. A. Davis, G. Musmann, I. Richter, C. Othmer, K. H. Glassmeier, S. Moses, Ion propulsion subsystem environmental effects on Deep Space One: Initial results from the IPS diagnostic subsystem, DS1 Technology Validation Report, NASA-JPL, 2000.

Buneman, O., T. Neubert, and K. I. Nishikawa, Solar wind-magnetosphere interaction as simulated by a 3-D EM particle code, *IEEE Trans. Plas. Sci.*, **20** (6), 810, 1992.

Buneman, O., TRISTAN: The 3-D E-M Particle Code, in: Matsumoto, H., and Y. Omura (Eds.), Computer Space Plasma Physics, Terrapub, Tokyo, p. 67, 1993.

Buneman, O., and G. Kooyers, Computer simulation of the electron mixing mechanism in ion propulsion, *AIAA-Journal*, **1**, 2525, 1963.

Buneman, O., and W. B. Pardo, Relativistic Plasmas, Benjamin, New York, p. 205, 1968.

Burger, P., Nonexistence of dc states in low-pressure thermionic converters, *J. Appl. Phys.*, **35**, 3048, 1964.

Carretti, E., Numerical simulations and power spectrum analysis of ESP redshift survey, PhD thesis, Università degli Studi di Bologna, Italy, 1998.

Chen, F. F., Plasma physics and controlled fusion, Plenum Press, New York, 1974.

de Boer, P. C. T., Measurements of electron density in the charge exchange plasma of an ion thruster, *J. Propulsion and Power*, **13**, 783, 1997.

Dunn, D. A., and I. T. Ho, Computer experiments on ion beam neutralization with initially cold electrons, *AIAA-Journal*, **1**, 2770, 1963.

Ferraro, R. D., P. C. Liewer, and V. K. Decyk, Dynamic load balancing for a 2D concurrent plasma PIC code, *J. Comput. Phys.*, **109**, 329, 1993.

Gustafson, J. L., Reevaluating Amdahl's Law, *Comm. Assoc. Comp. Mach.*, **31** (5), p. 532, 1988.

Harlow, F. H., The particle-in-cell computing method for fluid dynamics, in: Alder, B., S. Fernbach, and M. Rotenberg (Eds.), Methods of Computational Physics, **3**, 1964.

Hasegawa, A., and T. Sato, Existence of a negative potential solitary-wave structure and formation of a double layer, *Phys. Fluids*, **25**, 632, 1982.

Hockney, R. W., A fast direct solution of Poisson's equation using Fourier analysis, *J. Assoc. Comput. Mach.*, **12**, 95, 1965.

Hockney, R. W., and J. W. Eastwood, Computer Simulation Using Particles, New York, McGraw-Hill, 1981.

Hudson, M. K., and D. W. Potter, Electrostatic shocks in the auroral magnetosphere, in: Physics of auroral arc formation, American Geophysical Union, Washington D.C., p. 260, 1981.

Kittel, Ch., Einführung in die Festkörperphysik, Oldenbourg, München, 1999.

Liewer, P. C., and V. K. Decyk, A general concurrent algorithm for plasma particle-in-cell simulation codes, *J. Comput. Phys.*, **85**, 302, 1989.

Lindman, E., "Free-space" boundary conditions for the time dependent wave equation, *J. Comput. Phys.*, **18**, 66, 1975.

Lyster, P. M., P. C. Liewer, V. K. Decyk, and R. D. Ferraro, Implementation and characterization of three-dimensional particle-in-cell codes on multiple-instruction-multiple-data massively parallel supercomputers, *Computers in Physics*, **9** (4), 420, 1995.

Marder, B., A method for incorporating Gauss' law into electromagnetic PIC codes, *J. Comput. Phys.*, **68**, 48, 1987.

Matsumoto, H., and Y. Omura, Particle simulation of electromagnetic waves and its application to space plasmas, in: Matsumoto, H., and T. Sato (Eds.), Computer Simulation of Space Plasmas, Terrapub, Tokyo, p. 43, 1985.

Morse, R.L. and C.W. Nielson, Numerical simulation of the Weibel instability in one and two dimensions, *Phys. Fluids*, **14**, 830, 1971.

Minty, E., MPI-2: Extending the Message Passing Interface, Edinburgh Parallel Computing Centre, http://www.epcc.ed.ac.uk/epcc-tec/documents, 1998.

MPI, A Message-Passing Interface Standard, Message Passing Interface Forum, http://www.mpi-forum.org, 1994.

MPI-2, Extensions to the Message-Passing Interface, Message Passing Interface Forum, http://www.mpi-forum.org, 1997.

Nishikawa, K.I., Particle entry into the magnetosphere with a southward interplanetary magnetic field studied by a three-dimensional electromagnetic particle code, *J. Geophys. Res.*, **102**, 17631, 1997.

Nishikawa, K.I., Particle entry through reconnection grooves in the magnetopause with a dawnward IMF as simulated by a 3-D EM particle code, *Geophys. Res. Lett.*, **25**, 1609, 1998.

Okuda, H., Nonphysical noises and instabilities in plasma simulation due to a spatial grid, *J. Comput. Phys.*, **10**, 475, 1972.

Pierce, J.R., Limiting stable current in electron beams in the presence of ions, *J. Appl. Phys.*, **15**, 721, 1944.

Richter, I., FGM analysis for Deep Space 1, Long time data investigation, DS1-IGM-TR0004, Institut für Geophysik und Meteorologie, TU Braunschweig, 2000.

Richter, I., FGM analysis for Deep Space 1, The S-Peak operation, DS1-IGM-TR0008, Institut für Geophysik und Meteorologie, TU Braunschweig, 2001.

Richter, I., D.E. Brinza, M. Cassel, K.H. Glassmeier, F. Kuhnke, G. Musmann, C. Othmer, K. Schwingenschuh, and B.T. Tsurutani, First Direct Magnetic Field Measurements of an Asteroidal Magnetic Field: DS1 at Braille, *Geophys. Res. Lett.*, **28** (10), 1913, 2001.

Sagdeev, R.S., Plasmaphysik für Physiker, Teubner, Stuttgart, p. 329, 1979.

Samanta Roy, R., D. Hastings, and N. Gatsonis, Ion-thruster modeling for backflow contamination, *J. Spacecraft and Rockets*, **33** (4), 525, 1996a.

Samanta Roy, R., D. Hastings, and N. Gatsonis, Numerical study of spacecraft contamination and interaction by ion-thruster effluents, *J. Spacecraft and Rockets*, **33**(4), 535, 1996b.

Schüle, J., Parallel Computing with Emphasis on Distributed Systems, Shaker Verlag Aachen, 2000.

Tajima, T., Computational Plasma Physics, Addison-Wesley, New York, 1989.

Tajima, T., and Y. C. Lee, Absorbing boundary condition and Budden turning point technique for electromagnetic plasma simulations, *J. Comput. Phys.*, **42**, 406, 1981.

Tartz, M., E. Hartmann, R. Deltschew, and H. Neumann, Validation of a grid erosion simulation by short-time erosion measurements, *IEPC 99-147*, 26th Int. Electr. Prop. Conf., Kitakyushu, Japan, 1999.

Trivelpiece, A. W., and R. W. Gould, Space charge waves in cylindrical plasma columns, *J. Appl. Phys.*, **30**, 1784, 1959.

Villasenor, J., and O. Buneman, Rigorous charge conservation for local electromagnetic field solvers, *Comp. Phys. Comm.*, **69**, 306, 1992.

Wang, J., priv. commun., 2001.

Wang, J., and D. Hastings, Ionospheric plasma flow over large high-voltage space platforms II: The formation and structure of plasma wake, *Phys. Fluids*, **4** (6), 1615, 1992.

Wang, J., and J. Brophy, 3-D Monte-Carlo particle-in-cell simulations of ion thruster plasma interactions, *AIAA 95-2826*, 1995.

Wang, J., P. Liewer, and V. Decyk, 3D electromagnetic plasma particle simulations on a MIMD parallel computer, *Comp. Phys. Comm.*, **87**, 35, 1995.

Wang, J., J. Brophy, and D. Brinza, 3-D simulations of NSTAR ion thruster plasma environment, *AIAA 96-3202*, 1996.

Wang, J., D. E. Brinza, D. T. Young, J. E. Nordholt, J. E. Polk, M. D. Henry, R. Goldstein, J. J. Henly, D. J. Lawrence, and M. Shappirio, Deep Space One investigations of ion propulsion plasma environment: initial results, submitted to *J. Spacecraft and Rockets*, 2000.

Wadhwa, R. P. , O. Buneman, and D. F. Brauch, Two-dimensional computer experiments on ion-beam neutralization, *AIAA-Journal*, **3**, 1076, 1965.

Winske, D., and N. Omidi, A non-specialist's guide to kinetic simulations of space plasmas, *J. Geophys. Res.*, **101**, 17287, 1996.

Yee, K. S., Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media, *IEEE Trans. Ant. Propagat.*, **14**, 302, 1966.

Danksagung

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Geophysik und Meteorologie der Technischen Universität Braunschweig.

Mein besonderer Dank gilt Herrn Prof. Dr. Glaßmeier. Er betreute mich vorzüglich und räumte mir bei der Ausrichtung der Arbeit viele Freiheiten ein. Nicht nur in wissenschaftlicher Hinsicht stand er mir stets mit wertvollen Ratschlägen zur Seite. Mit seiner jederzeit offenen und bestärkenden Haltung ermöglichte er mir neben vielen nationalen und internationalen Tagungen den Besuch der International Space University und einen Forschungsaufenthalt am Edinburgh Parallel Computing Centre – zwei “Highlights” meiner Promotion.

Herrn Prof. Dr. Motschmann, auf den die Anregung zur Untersuchung von Ionentriebwerken zurückgeht, danke ich für sein großes Interesse an meiner Arbeit und seine fortwährende Unterstützung, speziell bei numerischen Problemen. Die erhellenden Gespräche, für die er sich stets viel Zeit nahm, hatten einen prägenden Einfluss auf die vorliegende Arbeit.

Spezieller Dank gebührt auch Dr. Josef Schüle und Christian Frick vom Rechenzentrum der TU Braunschweig, die durch ihr Engagement und ihre Expertise die Parallelisierung des Simulations-Codes zu einem der reizvollsten Abschnitte meiner Promotion machten.

Danken möchte ich ferner dem TRACS Team des Edinburgh Parallel Computing Centre für die freundliche Atmosphäre während meines dortigen Aufenthalts sowie Herrn Dr. Gavin Pringle für die fruchtbaren Diskussionen über die Optimierung des Simulations-Codes.

Allen jetzigen und ehemaligen Kollegen des Instituts für Geophysik und Meteorologie, die mit mir in irgendeiner Form zusammengearbeitet haben, danke ich für die angenehme und produktive Zeit am IGM. Stellvertretend sei hier die ertragreiche und kurzweilige Zusammenarbeit mit Dr. Ingo Richter erwähnt – sowohl im Projekt Deep Space 1 als auch bei unserer gemeinsamen “Leidenschaft”, der Kalibration von Satelliten-Magnetometern.

Ganz besonders danke ich meiner Frau Margarita, die mir während meiner Promotion den nötigen Rückhalt gab, sowie meinen Eltern und meiner Schwester für ihr stetes Interesse am Gelingen dieser Arbeit.